REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-02-

0123

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 26 FEB 02	3. RE FINAL (01 AUG 97 TO 31 JUL 01

4. TITLE AND SUBTITLE
HEALTH MONITORING ON VIBRATION SIGNATURES

5. FUNDING NUMBERS
F49620-98-1-0049

6. AUTHOR(S)
PROFESSOR GARY YEN

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
OKLAHOMA STATE UNIVERSITY
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
202 ENGINEERING SOUTH
STILLWATER, OK 74078-5032

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
801 N. RANDOLPH STREET
ARLINGTON, VA 22203

10. SPONSORING/MONITORING

20020402 080

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This final report covers the entire contracting effort from August 1, 1997 to July 31, 2001 (with an extended year at no cost tc the program). The progress made is primarily targeted to establish the fundamental bases from analytical and simulation studies. In addition, we have moved forward to validate the algorithms developed in several experimental testbeds. To consolidate the technical efforts dedicated to the theoretical developments appropriate for vibration monitoring in Air Force structures (e.g., aircraft, rotorcraft and RLVs), a comprehensive "modular" approach is taken. The tasks as outlined include Wavelet feature extraction, sensor validation, feature selection, fault detection, identification and classification, knowledge representation, multi-model on-line control autonomy, nonlinear system identification, adaptive critic fault tolerant control anc multi objective optimization design. The approach is evolving as more mathematical analyses are accomplished and desired specifications are defined. The methodology proposed is generic and applicable to a wide variety of industrial applications (e.g., chemical fluidized processes, semiconductor RIE facility), medical-assisted monitoring (e.g., EKG, intra-cranial pressure monitoring) and environmental assessment (e.g., frog/bird call monitoring). The vibration data set used for preliminary test is derived from U.S. Navy CH-46E Chinook helicopters available in public domain; commonly know as the Westland data set. The data set (4-second time series) is available in public domain and has been used for benchmark comparison by researchers within this community. The data set is clean and complete as opposed to the USAF Academy bearing data set, which is highly noisy and sparse.

14. SUBJECT TERMS

15. NUMBER OF PAGES
220

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	

Final Report:

HEALTH MONITORING ON VIBRATION SIGNATURES

Gary G. Yen, Associate Professor

Oklahoma State University
School of Electrical and Computer Engineering
202 Engineering South
Stillwater, OK 74078-5032

(Phone) 405-744-7743
(Fax) 405-744-9198
(E-mail) gyen@ceat.okstate.edu
(WWW) http://www.okstate.edu/elec-engr/faculty/yen

Grant Number: F49620-98-1-0049

Submitted to

Dr. Daniel Segalman
Air Force Office of Scientific Research
Directorate of Aerospace and Material Sciences
Structural Mechanics Program
110 Duncan Avenue, Suite B115
Bolling AFB, DC 20332-0001

1. OBJECTIVE

This DoD-EPSCoR research proposed to develop, design and evaluate an on-board intelligent health assessment tool for Air Force vibration monitoring applications. The system developed is capable of promptly detecting, correctly identifying, properly accommodating and reliably predicting the gradual material degradation and catastrophic component failures of Air Force vibrating structures in adverse operating environments. In addition, the system is equipped with the ability to reason the temporal cause-and-effect relationship from raw data for the purpose of predictive maintenance. In a similar spirit, the technology developed is spun-off to the monitoring and control of chemical processes, semiconductor equipment and environmental healthiness.

Adaptive variation of Wavelet packet transform is designed to extract necessary time-frequency signatures from analytically redundant sensor channels (Appendix A). To validate the healthiness of a given vibration sensor, neural network based observer is used to estimate critical sensor measurements when neighboring sensor readings are collected and collated (Appendix B). With the aid of statistical based feature selection criteria, many of the feature components containing little discriminant information have been discarded resulting in a feature subset having a reduced number of parameters without compromising the classification performance (Appendix C). The extracted reduced dimensional feature vector is then used as input to a pattern classifier. A hybrid neural/fuzzy network with an on-line real-time learning algorithm is then developed to perform intelligent decision making (Appendix D). To provide the functionality for predictive maintenance, knowledge representation and extraction is made possible through a fuzzy based reasoning (Appendix E). Additionally, two robust control laws based upon sliding mode variable structures and discrete-time Lyapunov stability theory are proposed to provide fault tolerance with guaranteed global stability and performance (Appendix F). To do so, a multi-model based nonlinear system identification based on Laguerre filters is suggested to formulate the changing dynamic during the evolution of failures (Appendix G). A hierarchical architecture that combines a high degree of reconfigurability and long-term memory is then proposed as a fault tolerant control algorithm for complex nonlinear systems (Appendix H). Dual Heuristic Programming is used for adapting to faults as they occur for the first time in an effort to prevent the build up of a general failure, and also as a tuning device after switching to a known scenario. A dynamical database, initialized with as much information of the plant as available, oversees the DHP controller. The decisions of which models to record, when to intervene and where to switch are autonomously taken based on specifically designed quality indexes. The problem formulation has resulted into a multiobjective optimization problem where a uniformly distributed, near optimal and near complete Pareto front is sought for (Appendix I) in the feedback loop of design procedure.

The resulted system with all needed components is then advocated to fulfill the time-critical and on-board needs in different levels of structural integrity over a global working envelope. The research dedicated to Air Force utilization is not only focused on mathematical treatments of the developed fault detection, identification and accommodation systems, but more importantly promote an ultimate enabling tool appropriate for on-board health decision making and adaptive control.

Use the technology developed, automatic recognition of frog vocalization is developed as a valuable tool for a variety of biological research and environmental monitoring applications (Appendix J). The simulation results show the promising future of deploying an array of continuous, on-line environmental monitoring systems based upon non-intrusive analysis of animal calls. In a similar spirit, we also propose a cost-effective and computation-efficient acoustic emission detection system combined with artificial neural network technology to recognize four major flow patterns in an air-water vertical two-phase vertical column (Appendix K). This technology is considered very crucial in most process, petrochemical and pharmaceutical industry.

The research objective is to demonstrate the feasibility and applicability of the proposed health monitoring procedures and reconfigurable control laws through mathematical analyses, numerical simulations and experimental verifications in chosen Air Force applications. The potential of spin-off applications on DoD structures (i.e., aeropropulsion engine, on-orbit satellite and reusable launch vehicle), industrial processes and environmental condition monitoring is promising and under pursuit. The research goal is in complement with building educational infrastructure, as witnessed by course development, laboratory institution, seminar organization and student training.

2. STATUS OF EFFORT

This final report covers the entire contracting efforts from August 1, 1997 to July 31, 2001 (with an extended year at no cost to the program). The progress made is primarily targeted to establish the fundamental bases from analytical and simulation studies. In addition, we have moved forward to validate the algorithms developed in several experimental testbeds. To consolidate the technical efforts dedicated to the theoretical developments appropriate for vibration monitoring in Air Force structures (e.g., aircraft, rotorcraft and RLVs), a comprehensive "modular" approach is taken. The tasks as outlined include Wavelet feature extraction, sensor validation, feature selection, fault detection, identification and classification, knowledge representation, multi-model on-line control autonomy, nonlinear system identification, adaptive critic fault tolerant control and multiobjective optimization design. The approach is evolving as more mathematical analyses are accomplished and desired specifications are defined. The methodology proposed is generic and applicable to a wide variety of industrial applications (e.g., chemical fludized processes, semiconductor RIE facility), medical-assisted monitoring (e.g., EKG, intra-cranial pressure monitoring) and environmental assessment (e.g., frog/bird call monitoring). The vibration data set used for preliminary test is derived from U.S. Navy CH-46E Chinook helicopters available in public domain, commonly known as the Westland data set. The data set (4-second time series) is available in public domain and has been used for benchmark comparison by researchers within this community. The data set is clean and complete as opposed to the USAF Academy bearing data set, which is highly noisy and sparse.

In addition, we have built a machinery fault simulator based on SpectraQuest motor test unit. This design allows us to simulate various real-time fault scenarios, including faulted bearing, shaft misalignment and imbalance of rotating inertia. The proof-of-the-concept system has been tested on much more complicated environments. The efforts dedicated to Wavelet packet feature extraction, sensor channel validation, statistical feature selection, neural-fuzzy fault classification, sliding mode fault tolerant control, multi-model nonlinear system identification, adaptive critic control and knowledge representation have been successful. The applications to Navy's Westland data and SpectraQuest machinery fault simulator show significant improvements from all available results given in literature. In addition, the Principal Investigator (PI) has continuously outreached to non-DoD community to spin-off the proven technology. The extension of using human-like sensory channels (i.e., acoustic emission sensor, CCD camera, omni-directional microphone and ultrasonic sensor) to monitor the chemical, pharmaceutical and petrochemical processes have been convincing. The extension to monitor environmental condition through frog or bird calls is in particular innovative. We continuously exchange information with POs at OKC-ALC, including B-1B, B-52, E-3T and C/KC-135 units to transfer the technology originally developed for rotorcraft to fix-wing aging aircraft. One contract has been finalized to develop the technology needed for capturing the failure modes information during the complicated maintenance actions.

The principal investigator would like to express his gratitude to Air Force Office of Scientific Research (AFOSR) and to program managers, Major Brian Sanders and Dr. Daniel Segalman for their generous support during the course of this study. Without their gracious financial and technical support, this research can never become feasible.

3. ACCOMPLISHMENTS

3.1 Introduction

Modern engineering technology is leading to increasingly complex Air Force vehicles with ever more demanding performance criteria. Imminent needs in prolonging service life and mission readiness for global defense challenges call for an even higher standard in structural reliability. A downsized workforce, a declining development budget and the desire for a "better, cheaper and smarter" resolution have further complicated the risk decisions. These problems are even crucial in today's global defense industry. Condition Monitoring has long been recognized as a top priority in the development of the next generation aircraft and reusable launch vehicles by BMDO. However, currently used diagnostic systems that rely primarily on ingenious sensor innovations or healthy redundant sensor placements to provide early warning and maintenance procedure are costly, vulnerable, labor-intensive and computationally expensive to validate.

In one extreme, the maintenance and sustainment of aging capital-intensive infrastructures demand innovative technology in condition-based maintenance. The USAF Aging Aircraft & Systems Office (ASC/AMA) located in Wright Patterson AFB has the direct responsibility within the Air Force components. A substantial and growing portion of the military transportation systems and infrastructure were built more than 40 years ago, and are now approaching or have exceeded the original design lifetime. An outstanding example of an aging aircraft is the U.S. Air Force's C/KC-135 airborne tanker fleet that is approximately 40 years of age and has been recently redesigned to remain in service at least through the year 2030. In a similar fate, MH-53J PAVE LOW helicopters at the U.S. Air Force Special Operations Command are fast approaching its destined lifetime and need to be on duty for another 40 years.

A 1997 study conducted by the National Research Council for the USAF addressed the technological status, needs and opportunities facing the Air Force's aging aircraft fleet. Of particular concern is the C/KC-135 tanker. Replacement cost for the C/KC-135 fleet alone is estimated at $40 billion, and this represents but one of the USAF aging systems. The OKlahoma City Air Logistic Center at Tinker AFB (OKC-ALC) is served as the primary aging aircraft maintenance depot for the USAF fleets of bomber (B-52 in service since 1961), tanker (C/KC-135 in service since 1958) and surveillance aircraft (E-3 AWACS in service since 1977). The PI and collaborative researchers in the State of Oklahoma have established the *Oklahoma Center for Aging Systems and Infrastructur* (OCASI) in providing the basic research, technology transfer and training to the OKC-ALC operations.

The OCASI is dedicated to developing technologies and analysis tools for predicting, extending and controlling the lives of systems and components of infrastructure such as airframes, roads and bridges, oil fields and refinery machinery, general physical plant and electrical/electronic equipment. As a founding member of OCASI, the PI has actively contributed in meeting the technology and education demands in facilitating the maintenance operation exercised in the OKC-ALC. The PI has discussed the potential opportunity with several POs at OKC-ALC to apply the technology developed within this program to the fixed-wing aircraft. In addition, the delegates from USAF Aging Aircraft and Systems Office, Boeing-Wichita, American Airlines and others are involved in the executive panel at OCASI.

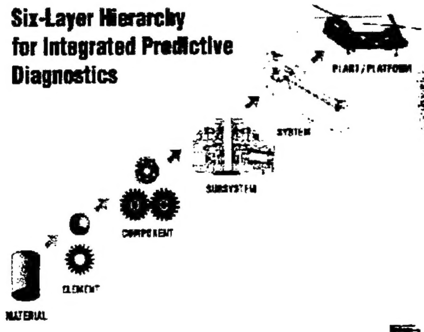
In the most recent board meeting (June 21, 2001), the PI presented the research undertaken herein supported by the AFOSR/NA to researchers from Hughes and Raytheon. Future collaboration is under discussion. Significant interests have been received from the Air Force end users and industrial technology developers. Continuous contacts and interactions with Air Force program offices and commercialization partners will be maintained to explore the future research opportunities. The potential and commitment of follow-up spin-off research from DoD end users and defense industry is developing.

This final report documents the progress made throughout the entire contracting period dedicated into the Wavelet packet feature extraction, sensor channel validation, statistical feature selection, neural-fuzzy classification, sliding mode fault tolerant control, multi-model nonlinear system identification, adaptive critic dynamic programming, knowledge representation and multiobjective optimization design.

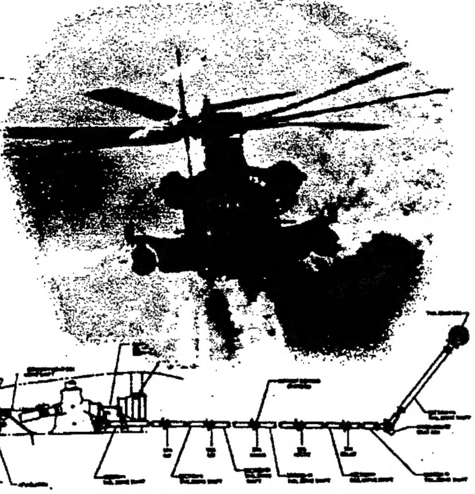
As shown in Figure 1, a generic, structure health monitoring approach is constantly evolving. The system assumes a given sensor suite will act as an on-line health usage monitor and at best provide the real-time control autonomy. The sensor suite can incorporate various types of sensory devices, from vibration accelerometers, omni-directional microphones, computer vision CCDs, pressure gauges to temperature indicators. The decision can be shown in a visual on-board display (for pilot or for ground maintenance crew) or fed to the control block to invoke controller reconfiguration. The approach has been continuously refined as more mathematical analyses are accomplished and desired specifications are defined.

VIBRATION HEALTH MONITORING
for Rotorcraft Wings

Six-Layer Hierarchy for Integrated Predictive Diagnostics

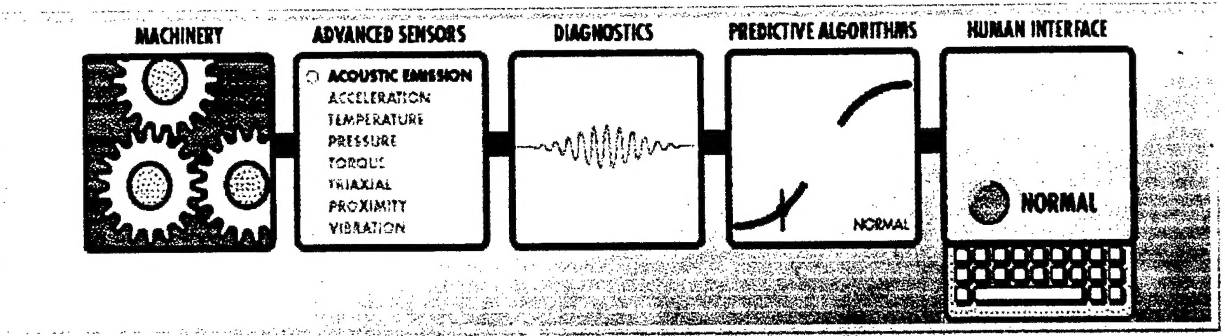


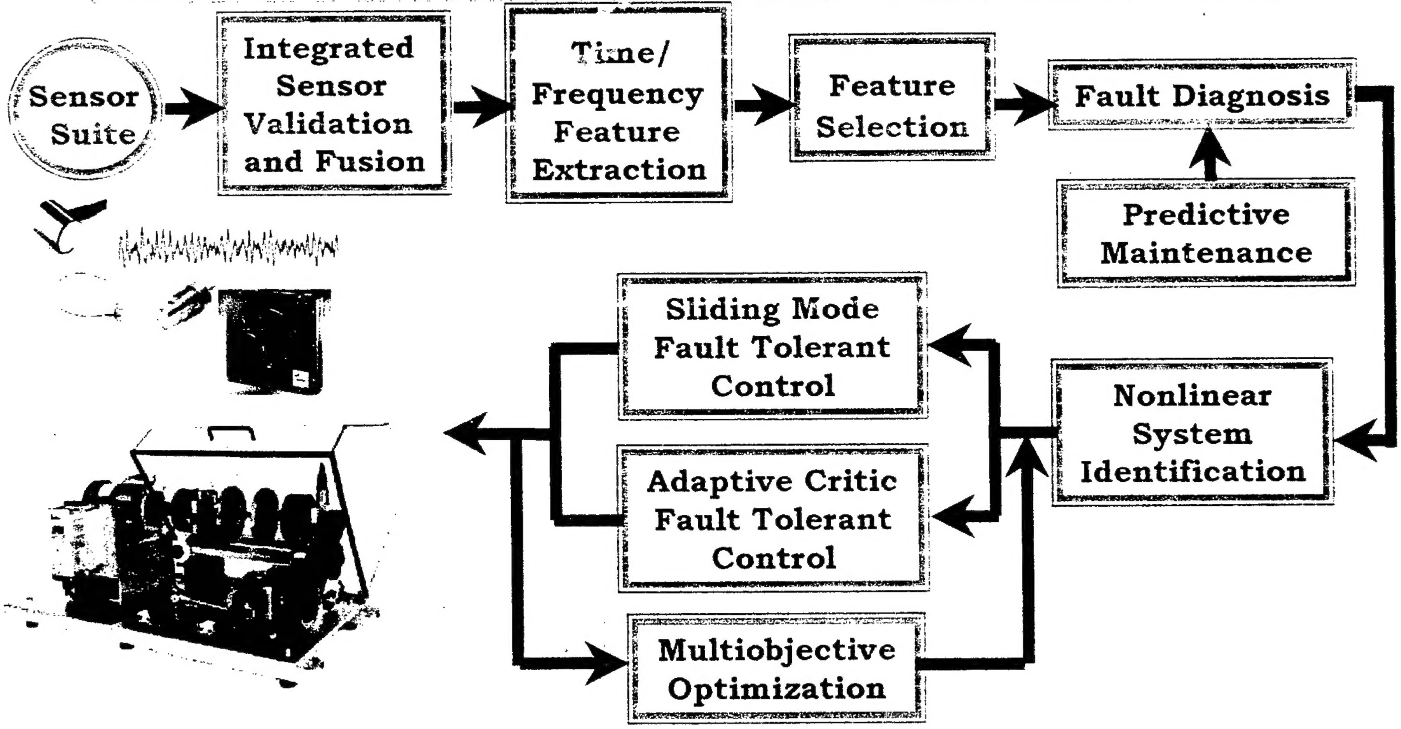
MH-53J PaveLow



TECHNICAL OBJECTIVES

This project features to design and evaluate an on-board intelligent health assessment tool for Air Force applications. The system developed is capable of on-line detecting, identifying, accommodating and predicting the gradual material degradation and catastrophic component failures of Air Force smart structures in an adverse operating environment.




Sensor Suite
Integrated Sensor Validation and Fusion
Time/ Frequency Feature Extraction
Feature Selection
Fault Diagnosis
Predictive Maintenance
Nonlinear System Identification
Sliding Mode Fault Tolerant Control
Adaptive Critic Fault Tolerant Control
Multiobjective Optimization

3.2 Wavelet Packet Feature Extraction

Condition monitoring of dynamic systems based on vibration signatures has generally relied upon Fourier based analysis as a means of translating vibration signals in the time domain into the frequency domain. However, Fourier analysis provided a poor representation of signals well localized in time. In this case, it is difficult to detect and identify the signal pattern from the expansion coefficients because the information is diluted across the whole basis. The Wavelet Packet Transform (WPT) is introduced as an alternative means of extracting time-frequency information from vibration signature. The resulting wavelet packet transform coefficients provide one with arbitrary time-frequency resolution of a signal. With the aid of statistical based feature selection criteria, many of the feature components containing little discriminant information could be discarded resulting in a feature subset having a reduced number of parameters without compromising the classification performance. The extracted reduced dimensional feature vector is then used as input to a neural network classifier. This has significantly reduced the long training time that is often associated with the neural network classifier and improved its generalization capability.

This research has investigated the feasibility of applying the wavelet packet transform to the classification of vibration signals. Using the wavelet packet transform, a rich collection of time-frequency characteristics in a signal can be obtained and examined for classification purposes. In this study we detailed an innovative feature selection process that exploits signal class differences in the wavelet packet node energy. This result in a reduced dimensional feature space compared to the dimension of the original time series signal. The wavelet packet based features, obtained by our method for vibration signals, yields nearly 100% correct classification when used as input to a neural network classifier.

Please refer to Appendix A for a technical report published in the *IEEE Transactions on Industrial Electronic.*

3.3 Sensor Channel Validation

The validation of data from sensors has become an important part in the operation and control of modern industrial equipment. To validate a signal, the sensor must be shown to consistently provide the correct data, and the analysis of the validation hardware or software should provide an alarm when the sensor signal deviates from its nominal value. Neural networks based models can be used to estimate critical sensor values when neighboring sensor measurements are used as inputs. The discrepancy between the measured and predicted sensor value may then be used as an indication of sensor health.

A methodology for estimating sensor values and detecting sensor failure has been developed in this work. The method allows us to estimate a critical sensor data when other sensors measurements are used as inputs. An auto-correlation analysis in frequency domain was used to detect the sensor failure. The network is a synergetic combination of fuzzy logic and neural networks. It employs the fast parallel computation and learning capability of neural networks. In addition, fuzzy set theory adds the ability to represent and manipulate imprecise information. The Winner-Take-All (WTA) Experts Networks consists of two main layers: Fuzzy membership cluster layer and MLP experts layer. The cluster layer employs the Gaussian radial basis function as a fuzzy membership function. The general idea is to divide a complicated problem into a series of sub-problems and assign a set of function approximators to each sub-problem. A growing fuzzy membership clustering methods was used to divide the input space into overlapping regions on which 'experts' act. After the WTA Experts Networks were trained in sensor nominal state, the estimation result was compared with real sensor outputs in failure modes. The difference between the two signals in frequency domain was calculated. The auto-correlation of the residual is analyzed to decide whether it is a white noise or not. Additionally, the jump indicator, mean indicator and variance indicator helped to identify sensor failures in time domain. The results both from frequency and time domain were combined together covering the eight failure modes known in literature.

Two benchmark data set: the Spectra Quest Machinery Fault Simulator data set and the Westland vibration data set were used in simulation experiments to demonstrate the performance of the WTA Experts Networks. Comparisons between the WTA Experts Networks and the other two neural networks estimators were made. The results show that, in terms of estimation performance (MSE), the WTA is competitive with or even better than the MLP networks and RBF networks alone. Furthermore, the auto-correlation analysis based sensor validation algorithm was used to investigate eight sensor failure modes in frequency domain. The results from the simulation studies have shown that the validation algorithm is efficient for detection seven of the eight faults, except the 'Spike' mode. With the help of indicators in time domain, the detection algorithm covered all eight faults at last.

Please refer to Appendix B for a technical report published in the *ISA Transactions*.

3.4 Statistical Feature Selection

One advantage of using wavelet packets transform to decompose a signal is that it allows us to examine different time-frequency resolution components in a signal. However, direct manipulation of a whole set of node energies is prohibitive because the space normally has very high dimensionality, and the existence of undesired components makes the classification unnecessarily difficult. In the training of a neural network classifier, it is desirable to use a lower dimensional vector as input to the neural network to ease the design of the classifier and improve its generalization capability. One popular technique in reducing the feature dimensionality is the Karhumen-Loéve (K-L) transform. The K-L transform is optimal for "*signal representation*" in the sense that it provides the smallest mean square error for a given number of data. However, the features defined by the K-L transforms are not optimal for "*class separability*". One transformation associated with this assumption is based on within and between class scatter matrices that are used in linear discriminant analysis of statistics. The idea is to find a linear transformation that projects the samples onto a lower dimensional space in which the variability of samples within each class is as close as possible, and the dispersion of the class mean vectors about the mean vector is as separated as possible. In such a case, two feature selection criteria based on measures of the overlap of the conditional probability density function among different classes was proposed to avoid the possible numerical problem.

Please refer to Appendix C for a technical report published in the *ISA Transactions*.

3.5 Neural-Fuzzy Fault Classification

An innovative neuro-fuzzy system (interconnecting architecture and learning rule) appropriate for fault detection, identification and classification in a machinery condition health monitoring environment, called an "Incremental Learning Fuzzy Neural" (ILFN) network is proposed. The ILFN classifier, using localized neurons to represent the distributions of the input space, is a fast, real-time, one-pass, on-line and incremental learning algorithm. The ILFN network employs a hybrid supervised and unsupervised learning scheme to generate its prototypes. The network is a self-organized classifier with the ability to adaptively learn new classes of failure symptoms without forgetting existing knowledge. The classifier can detect new classes of failure modes and update its parameters continuously while monitoring a system. To demonstrate the feasibility and effectiveness of the proposed neuro-fuzzy paradigm, numerical simulations have been performed using the vibration data known as Westland data set collected from an U.S. Navy CH-46E helicopter teststand. Using a simple fast Fourier transform technique for feature extraction, the ILFN network capable of one-pass, on-line and incremental learning has shown promising results. With various torque levels for training the network, 100% correct classification was achieved for the same torque levels of testing data. In addition, the classification performance of the network has been tested on some well-known benchmark data sets, such as the Fisher's Iris data and the Deterding vowel data set. For the generalization capability, comparison studies with other well-known classifiers were performed and the ILFN classifier was found competitive with or even superior to many existing classifiers.

Please refer to Appendix D for a technical report published in the *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics.*

3.6 Fuzzy Knowledge Representation

Among all well-known failure modes, an experienced expert can usually tell that an erratic frequency response of a bearing sensor may indicate the degradation of the transmission gearbox. An experienced operator in sulfuric acid treatment of phosphate rock may observe froth color or bubble character to control process material in-flow. The capability of incorporating this abstract expert knowledge into a learnable artificial neural network becomes essential in solving the FDIA problem effectively. The existing feedforward networks, which learn and generalize all nonlinear mappings from raw data, does not assure such a mechanism.

Knowledge representation is the ability to translate knowledge entailed in a "black box" neural network structure into linguistic and/or numeric rules that are accessible to human system operators. The two main functions of an extracted rule set is to clearly explain knowledge provided by domain experts and to provide a means for human operators to validate the operation of the neural network. Rules provide reasoning and explanation capabilities for the system. The classification output of the neural network can be justified in linguistic terms, increasing human understanding of the logical process used by the network and identifying any mis-classifications that are known to be incorrect by domain experts. Classifications made by the neural network can be verified by the knowledge base by examining the results of presented input patterns to the network and the rule base. Linguistic rules also help alleviate the interference problem of artificial neural networks. When a neural network is trained using data from the same system in different operating environments, a set of rules for each environment may be extracted. This simplifies retraining, and allows portability of the network and its rules between systems and environments.

In this study, we propose a novel hybrid intelligent system (HIS), which provides a unified integration of numerical and linguistic knowledge representations. The proposed HIS is a hierarchical integration of an incremental learning fuzzy neural network (ILFN) and a linguistic model, i.e., fuzzy expert system (FES), optimized via the genetic algorithm (GA). The ILFN is a self-organizing network with the capability of fast, one-pass, online, and incremental learning. The linguistic model is constructed based on knowledge embedded in the trained ILFN or provided by the domain expert. The knowledge captured from the low-level ILFN can be mapped to the higher-level linguistic model and vice versa. The GA is applied to optimize the linguistic model to maintain high accuracy, comprehensibility, completeness, compactness, and consistency. The resulted HIS is capable of dealing with low-level numerical computation and higher-level linguistic computation. After the system being completely constructed, it can incrementally learn new information in both numerical and linguistic forms. To evaluate the system's performance, the well-known benchmark Wisconsin breast cancer data set was studied for an application to medical diagnosis. The simulation results have shown that the proposed HIS perform better than the individual standalone systems. The comparison results show that the linguistic rules extracted are competitive with or even superior to some well-known methods.

Please refer to Appendix E for a technical report submitted to the *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*.

3.7 Sliding Mode Fault Tolerant Control

As dynamic systems become more complex, experience more rapidly changing environments, and encounter a greater variety of unexpected component failures, solving the control problems of such systems is a grand challenge for control engineers. Traditional control design techniques are not adequate to cope with these systems, which may suffer from unanticipated dynamic failures. In this research work, we investigate the fault tolerant control problem, the current existing intelligent control techniques using artificial neural networks, and propose an intelligent control strategy to handle the desired trajectories tracking problem for systems suffering from catastrophic faults or incipient failures. The approach is to continuously monitor the system performance and identify what the system's current state is by using a fault detection method based upon our best knowledge of the nominal system and nominal controller. Once a fault is detected, the proposed intelligent controller will adjust its control signal by adding a robust term (i.e., by switching to a sliding mode controller) to confine the system performance within a boundary layer. At the same time, an artificial neural network is initialized and compensates for the unknown fault dynamics on-line. Once the on-line learning process converges, the control input is tuned again by using the output of the identification model and a new least upper bound for the remaining uncertainty to further reduce the tracking error. The simulation results show a significant improvement in trajectory following performance based upon the proposed *intelligent* sliding mode controller.

We investigated fault tolerant control problems and proposed an intelligent sliding mode control strategy to deal with a specific fault tolerant control problem based upon the discrete-time sliding mode control technique and neural network on-line approximator that is capable of *self-optimization, on-line adaptation, autonomous fault detection* and *controller reconfiguration.*

We, first, suggest using neural network techniques to improve the accuracy of the nominal model and the nominal controller. Then, the abnormal system behavior due to system component failures can be monitored and identified. Once system faults are detected, the control law is reconfigured by using sliding mode control technique together with the help of a neural network to recover the system performance. The neural network is used to learn the unknown failure dynamics on-line and to estimate the least upper bound of the remaining uncertainty. The resulting intelligent control law shows promising performance. The simulation results indicate a significant performance improvement based upon the proposed intelligent control strategy even when the system becomes unstable, due to multiple unexpected component failures. Further research work will focus on developing an intelligent control methodology for more general failure cases where both the nominal model and the unknown failure dynamics are general nonlinear functions. When the nominal model is not in "affine in control" format or it is not readily available to obtain the mathematical representation, we may use neural networks to replace it. Under this structure, the problems of learning the unanticipated failure mode dynamics on-line and the reconfiguration of the control actions are much more difficult to solve in a real-time fashion.

Please refer to Appendix F for a technical report to be published in the *International Journal of Control.*

3.8 Multi-Model Nonlinear System Identification

Recently, model-free or data-driven control has been gaining a great interest to overcome the limitations of the conventional model based control methodologies. However, the existing data-driven control is far from practical because of its slow convergence, severe computational burden and lack of analysis and synthesis tools. This paper was motivated on response to these issues of data-driven control methodologies by the multiple model approach.

First, a feasibility check of the data-driven approach is done. Since we do not make any assumptions about the system, it is important to verify that the unknown system can be identified only with sampled input-output data. After the discussion about the embedding theorem, literature is reviewed regarding nonlinear sampled data system identification, multiple modeling and multiple model based control. Literature review reveals that the multiple model approach is quite promising, however, it still lacks systematic tools for analysis and synthesis. A new approach based on orthonormal bases is proposed to maintain the tractability while keeping the advantages of multiple models. Simulation study is included to verify the existing as well as new algorithms. Conclusions and new suggestions about improvement are followed.

This paper was motivated by the ambition to realize a practical data-driven control system comparable to conventional model based methods. The main theme was to adopt a multiple model approach instead of a global approach. By this adoption, we can relieve the computational burden significantly while also maintain the mathematical tractability. Also. this approach enables us to take advantage of the existing methods such as linear system identification and many estimation techniques. The proposed algorithm takes advantage of the ice property of orthonormal basis functions. By this, we can relieve the difficulty of estimating the order of regression vectors and also achieve efficient training with maintained mathematical tractability. The simulation results in section 4 verified the algorithm. Three other models were also considered: linear ARX models, feedforward neural networks based ARX models and SOM based multiple models. The simulation was done with Matlab and a small toolbox was written for this study. The proposed model needs more refinement and analysis. One issue is to select the proper weights. One possible idea is to utilize the correlation since the local models are linear systems and we assume that the nonlinear system is locally linear. Another issue is locating the optimal poles. As mentioned before, there are many advantages if we can make the Laguerre bases orthonormal to each other. Also. the pole estimation method based on the iterative optimization method has limitations in on-line implementation.

Please refer to Appendix G for a technical report to be published in the *International Journal of Control.*

3.9 Adaptive Critic Dynamic Programming

As complex systems suffer from faults, the original model parameters, or even their own dynamic structure, may change in a multitude of unpredictable ways. Even if the system has a satisfactory linearization around the nominal operating point, nonlinearities may become of paramount importance after a fault occurs. Since complex systems pose a challenge even in the design of models under nominal conditions, the task of off-line devising nonlinear high order models for all known fault scenarios can be a daunting one. When the stochastic nature of faults is taken into consideration, and to even possess knowledge of all fault scenarios is made impossible, it becomes clear to see that the problem of interest to FTC cannot be dealt with without on-line nonlinear adaptive control strategies. In the proposed architecture, Dual Heuristic Programming (DHP), an Adaptive Critic Design (ACD), was chosen as the reconfigurable controller due to its known effectiveness to work in noisy, nonlinear environments while making minimal assumptions regarding the nature of that environment.

To our best knowledge, the application of the DHP reconfigurable controller represents one of the most effective ways to deal with the unexpected dynamics that a plant may assume after the occurrence of a fault. However, as a FTC scheme by itself, the use of a reconfigurable controller such as DHP presents two main limitations. The first one arises from the fact that solutions to a set of expected fault scenarios are often available and may involve the application of very specific control laws. A reconfigurable controller alone however, does not provide any mechanism through which knowledge available during design time can be incorporated. The second limitation arises from the known tradeoff between adaptation and long-term memory. As the reconfigurable controller provides faster convergence to a wider range of control solutions, it fails to retain the knowledge of the control laws designed for previously visited scenarios.

To overcome both limitations, a novel supervisor system oversees the DHP controller in the architecture. The Identifier and Controller Dynamical Database, located inside the supervisor contains the knowledge available during design time, as well as solutions devised online for unexpected fault scenarios. The decisions of when to intervene by switching to a known control solution and when to add a new identifier and controller pair to the database are taken by the supervisor based on the current fault scenario. Such information is extracted by the scenario recognition module, which makes use of specifically designed quality indexes capable, not only of performing Fault Detection and Identification, but also to produce indispensable information on the evolution of a fault through time. The synergetic combination of the superior adaptation capabilities of the DHP controller with the fault information and long-term memory provided by the multiple model structure of the proposed supervisor generates an advanced FTC scheme capable to deal with a diversified collection of actuator and component faults.

Please refer to Appendix H for a technical report submitted to the *IEEE Transactions on Neural Networks*.

3.10 Multiobjective Optimization

Since the 1980's, the application of Evolutionary Algorithms (EA's) in solving Multiobjective Optimization Problems (MOPs) has been receiving a growing interest from evolutionary computation community. To search for a family of "acceptable" solutions, a so called Pareto set, by using EA's population-based parallel searching ability, several MultiObjective Evolutionary Algorithms (MOEAs) have been proposed. However, most of these MOEAs have difficulty in dealing with the trade-off between uniformly distributing the computational resources and finding the *near-complete* and *near-optimal* Pareto set. On the other hand, according to the No Free Lunch theorems, no formal assurance of an algorithm's general effectiveness exists if insufficient knowledge of the problem characteristics is incorporated into the algorithm domain.

In this study, the PI and his student propose a new evolutionary approach to multiobjective optimization problems, the Rank-Density based Genetic Algorithm (RDGA) that synergistically integrates selected features from existing MOEAs in a unique way. A new ranking method, automatic accumulated ranking strategy, and a "forbidden region" concept are introduced, completed by a revised adaptive cell density evaluation scheme and a rank-density based fitness assignment technique. In addition, four types of MOP features, such as discontinuous and concave Pareto front, local optimality, high-dimensional decision space and high-dimensional objective space are exploited and the corresponding MOP test functions are designed. By examining the selected performance indicators, RDGA is found to be statistically competitive with four state-of-the-art MOEAs in terms of keeping the diversity of the individuals along the trade-off surface, tending to extend the Pareto front to new areas and finding a well-approximated Pareto optimal front.

For the MOP test functions that only possess discontinuous or concave Pareto fronts, the recent developed approaches—RDGA, NSGA-II, PAES and SPEA-II do not have much trouble in finding some points of the true Pareto front, and RDGA is found to show better performance in keeping the diversity of the individuals along the current trade-off surface, extending the Pareto front to new areas, and finding a well-approximated, non-dominated set. However, without cautious selection of an initial population, an MOP with a feature of local optimality will easily cause most MOEAs problems in finding a pure global Pareto front. A local Pareto front created by constraints may produce less difficulty than what is generated by objective functions if the former one does not contain pseudo-global Pareto fronts. In addition, two complicated MOP test functions with high-dimensional decision space and objective space are examined by RDGA and the selected MOEAs. The experimental results demonstrate that RDGA produces statistically competitive results with other representative Pareto-based MOEAs in finding a *near-optimal*, *near-complete* and *uniformly distributed* Pareto front. Furthermore, as the test functions used in this study are still far from embodying a complete MOP test suite, a more profound study in developing a general model of MOEA and designing a more representative test function set in this field is absolutely necessary in future work.

Please refer to Appendix I for a technical report submitted to the *IEEE Transactions on Evolutionary Computations.*

3.11 Frog Call Monitoring

Recently there is an increasing interest and expenditure in environmental monitoring, both in North America and around the world. It is becoming essential to predict and assess the environmental impact of human activities on plants and animals. The populations of certain kinds of animals like birds and frogs are excellent indicators of overall environmental health. As many of the animals in an area may be heard but not seen, it is convenient to rely on their sounds as a means of identification. In many places manual census is not feasible, if not completely impossible. As a result, automatic recognition of animal sounds is considered a valuable tool for biological research and environmental monitoring applications.

In this research an automatic monitoring system, which can recognize the vocalizations of four popular species of frogs and can identify different individuals within the species of interest, is proposed. For the desired monitoring system, species identification is performed first with the proposed filtering and grouping algorithm. Individual identification, which can estimate frog population within the specific species, is performed in the second stage. Digital signal pre-processing, feature extraction, dimensionality reduction, and neural network pattern classification are performed step by step in this stage. Wavelet Packet feature extraction together with two different dimension reduction algorithms are synergistically integrated to produce final feature vectors, which are to be fed into a neural network classifier. The simulation results show the promising future of deploying an array of continuous, on-line environmental monitoring systems based upon non-intrusive analysis of animal calls.

Please refer to Appendix J for a technical report published in the *International Journal of Computational Intelligence and Applications.*

3.12 Acoustic Monitoring for Process Flow

Gas-liquid two-phase flows are widely used in the chemical industry. Accurate measurements of flow parameters, such as flow regimes, are the key of operating efficiency. Due to the interface complexity of a two-phase flow, it is very difficult to monitor and distinguish flow regimes on-line and real-time. In this paper we propose a cost-effective and computation-efficient AE detection system combined with artificial neural network technology to recognize four major patterns in an air-water vertical two-phase flow column. Several crucial AE parameters are explored and validated, and we found that the density of acoustic emission events and ring-down counts are two excellent indicators for the flow pattern recognition problems. Instead of the traditional Fair map, a hit-count map is developed and a multi-layer Perceptron neural network is designed as a decision-maker to describe an approximate transmission stage of a given two-phase flow system.

Please refer to Appendix K for a technical report to appear in the *ISA Transactions*.

4. PERSONNEL SUPPORT

4.1 Principal Investigator

Gary YEN, Associate Professor

4.2 Graduate Students

Liang-Wei HO, Ph.D. Fall 2000
sliding mode fault tolerant control
Nick LEE, Ph.D. Fall 2001
Laguerre-based multi-model system identification
Phayung MEESAD, Ph.D. Fall 2001 (supported by Thai Government)
knowledge representation and discovery
Haiming LU, Ph.D. candidate, to be completed in Spring 2002
multiobjective optimization
Pedro de LIMA, Ph.D. candidate, to be completed in Spring 2003
adaptive critic fault tolerant control

Kuo-Chung LIN, M.S. Fall 1998
Wavelet transform feature extraction
Wei FENG, M.S. Spring 2000
sensor data validation and fusion
Fengming YANG, M.S. Spring 2000
reinforcement learning control
Qiang FU, M.S. Fall 2000
environmental monitoring using frog vocalization

Please refer to Appendix L for the overview charts associate with each subtask outlined above.

5. PUBLICATIONS

5.1 Journal Article

- "Wavelet packet feature extraction for vibration monitoring," Yen G.G. and Lin K., *IEEE Transactions on Industrial Electronics*, Vol. 47, No. 3, June 2000, pp. 650-667.

- "Pattern classification by a neurofuzzy network: application to vibration monitoring," Meesad P. and Yen G.G., *ISA Transactions*, Vol. 39, No. 3, September 2000, pp. 293-308.

- "Winner take all expert network for sensor validation," Yen G.G. and Feng W., *ISA Transactions*, Vol. 40, No. 2, May 2001, pp. 99-110.

- "Automated frog-call monitoring system: a machine learning approach," Yen G.G. and Fu Q., *International Journal of Computational Intelligence and Applications*, Vol. 1, No. 2, June 2001, pp. 165-186.

- "An effective neural-fuzzy paradigm for machinery condition health monitoring," Yen G.G. and Meesad P., *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, Vol. 31, No. 4, August 2001, pp. 523-536.

- "Constructing a fuzzy rule-based system using the ILFN network and genetic algorithm," Yen G.G. and Meesad P., *International Journal of Neural Systems*, Vol. 11, No. 5, October 2001, pp. 427-443.

- "Acoustic emission data assisted process monitoring," Yen G.G. and Lu H., *ISA Transactions*, to appear.

- "Intelligent on-line fault tolerant control for unanticipated catastrophic failures," Yen G.G. and Ho L., *International Journal of Control*, to appear.

- "Hierarchical genetic algorithm based feed-forward neural network design," Yen G.G. and Lu H., *International Journal of Neurnl Systems*, to appear.

- "Identification of a deterministic constant-affine state space model with known initial conditions," Yen G.G. and Lee S., *IEE Proceedings- Control Theory and Applications*, to appear.

- "On-line multiple-model based fault diagnosis and accommodation," Yen G.G. and Ho L., submitted to *IEEE Transactions on Industrial Electronics* (under revision).

- "Rank and density-based genetic algorithm for multi-criterion optimization," Yen G.G. and Lu H., submitted to *IEEE Transactions on Evolutionary Computations* (under revision).

- "Reconfigurable control system design for fault diagnosis and accommodation," Ho L. and Yen G.G., submitted to *Asian Journal of Control* (under revision).

- "On-line fault accommodation control for catastrophic system failures," Ho L. and Yen G.G., submitted to *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*.

- "Multiple model approach by orthonormal bases for controller design," Yen G.G. and Lee S., submitted to *International Journal of Control and Intelligent Systems.*

- "On the local interpretation of Tgkagi-Sugeno fuzzy models from dynamical systems view," Lee S. and Yen G.G., submitted to *IEEE Transactions on Fuzzy Systems.*

- "Combined numerical and linguistic knowledge representations for medical diagnosis," Meesad P. and Yen G.G., submitted to *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics.*

- "Hierarchical rank-density genetic algorithm for radial basis function neural network design," Yen G.G. and Lu H., submitted to *International Journal of Computational Intelligence and Applications.*

- "Quantitative measure on the accuracy, comprehensibility, and completeness of a fuzzy expert system," Meesad P. and Yen G.G., submitted to *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics.*

- "Dynamic population size in multiobjective evolutionary algorithm," Lu H. and Yen G.G., submitted to *IEEE Transactions on Evolutionary Computations.*

- "Reinforcement learning algorithms for robotic navigation in dynamic environments," submitted to *International Journal of Computational Intelligence and Applications.*

- "Dynamic database approach to fault tolerant control using an adaptive critic design," Yen G.G. and Lima P.G., submitted to *IEEE Transactions on Neural Networks.*

5.2 Conference Proceedings

- "Health monitoring on vibration signatures- industrial applications," Yen G.G., *30th IEEE Southeastern Symposium on System Theory*, March 8-10, 1998, Morgantown, West Virginia.

- "Knowledge representation based on vibration monitoring," Yen G.G., *SPIE International Conference on Applications and Science of Computational Intelligence*, April 13-16, 1998, Orlando, Florida.

- "Sensory-based expert monitoring and control," Yen G.G., *SPIE International Conference on Applications and Science of Computational Intelligence II*, April 5-8, 1999, Orlando, Florida.

- "Sensory-based expert monitoring and control," Yen G.G., *2nd International Conference on Information Fusion*, July 6-8, 1999, Sunnyvale, California.

- "Wavelet packet feature extraction for fault detection, identification, and classification," Yen G.G. and Lin K., *1999 International Joint Conference on Neural Network*, July 10-16, 1999, Washington, District of Columbia.

- "Pattern classification by an incremental learning fuzzy neural network," Yen G.G. and Meesad P., *1999 International Joint Conference on Neural Network*, July 10-16, 1999, Washington, District of Columbia.

- "An effective neuro-fuzzy paradigm for machinery condition health monitoring," Yen G.G. and Meesad P., *1999 IEEE International Conference on Control Applications*, August 22-27, 1999, Kohala Coast Island, Hawaii.

- "Wavelet packet feature extraction for vibration monitoring," Yen G.G. and Lin K., *1999 IEEE International Conference on Control Applications*, August 22-27, 1999, Kohala Coast Isalnd, Hawaii.

- "Condition health monitoring using vibration signatures," Yen G.G. and Lin K., *38th IEEE Conference on Decision and Control*, December 7-10, 1999, Phoenix, Arizona.

- "Fault classification by a neurofuzzy network," Meesad P. and Yen G.G., *54th Meeting of the Society for Machinery Failure Prevention Technology*, May 1-4, 2000, Virginia Beach, Virginia.

- "Hierarchical genetic algorithm based multi-layer feedforward neural network design," Yen G.G. and Lu H., *1st IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, May 11-12, 2000, San Antonio, Texas.

- "Fault tolerant control: an intelligent sliding mode control strategy," Yen G.G. and Ho L., *2000 American Control Conference*, June 28-30, 2000, Chicago, Illinois.

- "Multiple model approach by orthonormal bases for controller design," Yen G.G. and Lee S., *2000 American Control Conference*, June 28-30, 2000, Chicago, Illinois.

- "Intelligent fault tolerant control using artificial neural networks," Yen G.G. and Ho L., *2000 IEEE/INNS International Joint Conference on Neural Network*, July 24-27, 2000, Como, Italy.

- "Winner take all expert network for sensor validation," Yen G.G. and Feng W., *2000 IEEE International Conference on Control Applications*, September 25-27, 2000, Anchorage, Alaska.

- "Constructing a fuzzy expert system using the ILFN network and the genetic algorithm," Yen G.G. and Meesad P., *2000 IEEE International Conference on Systems, Man, and Cybernetics*, October 8-11, 2000, Nashville, Tennessee.

- "Intelligent sensor validation by a hierarchical mixture of expert network," Yen G.G. and Feng W., *2000 IEEE International Conference on Industrial Electronics, Control, and Instrumentation*, October 22-28, 2000, Nagoya, Japan.

- "Development of a neuro-fuzzy expert system for predictive maintenance," Yen G.G. and Meesad P., *SPIE International Conference on Component and Systems Diagnostics, Prognosis, and Health Management*, April 16-17, 2001, Orlando, Florida.

- "On-line intelligent fault tolerant control for catastrophic system failures," Yen G.G. and Ho L., *SPIE International Conference on Component and Systems Diagnostics, Prognosis, and Health Management*, April 16-17, 2001, Orlando, Florida.

- "On-line intelligent fault accommodation control for catastrophic system failures," Ho L. and Yen G.G., *2001 American Control Conference*, June 25-27, 2001, Arlington, Virginia.

- "Reconfigurable control system design for fault diagnosis and accommodation," Ho L. and Yen G.G., *2000 INNS/IEEE International Joint Conference on Neural Networks*, July 14-19, 2001, Washington, District of Columbia.

- "A neurofuzzy network and its application to machine health monitoring," Meesad P. and Yen G.G., *2000 INNS/IEEE International Joint Conference on Neural Networks*, July 14-19, 2001, Washington, District of Columbia.

- "A hybrid intelligent system for medical diagnosis," Meesad P. and Yen G.G., *2000 INNS/IEEE International Joint Conference on Neural Networks*, July 14-19, 2001, Washington, District of Columbia.

- "Coordination of exploration and exploitation in a dynamic environment," Yen G.G., Yang F., Hickey T. and Goldstein M., *2000 INNS/IEEE International Joint Conference on Neural Networks*, July 14-19, 2001, Washington, District of Columbia.

- "A SOM mapping technique for visualizing documents in a database," Morris S., Wu Z. and Yen G.G., *2000 INNS/IEEE International Joint Conference on Neural Networks*, July 14-19, 2001, Washington, District of Columbia.

- "On-line multiple-model based fault diagnosis and accommodation," Yen G.G. and Ho L., *IEEE International Symposium on Intelligent Control*, September 5-7, 2001, Mexico City, Mexico.

- "Multiobjective optimization design via genetic algorihm," Lu H. and Yen G.G., *IEEE Conference on Control Applications*, September 5-7, 2001, Mexico City, Mexico.

- "Reconfigurable control system design for fault tolerance," Ho L. and Yen G.G., *40th IEEE Conference on Decision and Control*, December 4-7, 2001, Orlando, Florida.

6. INTERACTIONS/TRANSITIONS

6.1 Oklahoma City Air Logistic Center, Tinker AFB, OK

The PI is in the process to identify the problem issues that may be benefit from the technology developed within this program to OKC-ALC. The PO's at B-1B. B-52, C/KC-135 and E-3 are in the list.

6.2 IMC-Agrico, Mulberry, FL

The IMC-Agrico based on Mulberry, FL is interested in the sensory-based process monitoring, mainly on petroleum products. The research objective is to survey existing best practices in sensory-based process monitoring and control.

In daily practice, the experienced operator in sulfuric acid treatment of phosphate rock may observe froth color or bubble character to control process material in flow. The acoustic sound of cavitation or boiling/flashing may invoke the increase or decrease of material flow rates. Smart in-situ sensors, including proprietary low-power laser telemeter, surface photo voltage map, power spectrum of vibration signatures, ultrasonic/ultraviolet waveguides, machine vision, infrared telemetry and artificial nose/tongue have facilitated potential mechanism for factory automation with promising industry applicability. Based on the findings within DEPSCoR program, we are developing process specific health monitoring and control system. An array of error sensing microphones and a continuous stream video camera will be employed to pick up the acoustic sound or video image to invoke the intelligent decision making of a two-phase air-water flow distillation column.

6.3 Oklahoma Environmental Institute, Stillwater, OK

Automated environmental monitoring based solely upon frog vocalizations is considered a valuable tool for a variety of biological research and environmental monitoring indicators. We propose to develop an automated, unattended, environmental-harden, monitoring system, which can recognize the vocalization of interested species of frogs in the State of Oklahoma. The proposed monitoring system will deploy an omni-directional microphones array to record the frog calls in the field continuously, process the analog voices pick-ups (including background noise cancellation, signal conditioning, time-frequency feature extraction, data compression, and spectrogram signature classification) and then transmit only essential information over Mesonet for follow-up environmental decision making. The successful development of the proposed frog calls monitoring system will provide a robust measurement to quantify the environmental noise pollution. This proposal has been well-received by the Oklahoma City Zoo and North American Amphibian Society to monitor the amphibian population as an indicator of environmental and water quality.

7. PATENT DISCLOSURES

None

8. HONORS/AWARDS

Promoted to Associate Professor in
2000 Oklahoma State University Halliburton Outstanding Young Faculty Award
Best Presentation at 2000 American Control Conference, Chicago, IL, June 2000

APPENDIX A:

Wavelet Packet Feature Extraction For Vibration Monitoring

by

Gary G. Yen and Kuo-Chung Lin

IEEE Transactions on Industrial Electronics, **47**(3), 2000, pp. 650-667

Wavelet Packet Feature Extraction for Vibration Monitoring

Gary G. Yen, *Senior Member, IEEE,* and Kuo-Chung Lin

Abstract—**Condition monitoring of dynamic systems based on vibration signatures has generally relied upon Fourier-based analysis as a means of translating vibration signals in the time domain into the frequency domain. However, Fourier analysis provided a poor representation of signals well localized in time. In this case, it is difficult to detect and identify the signal pattern from the expansion coefficients because the information is diluted across the whole basis. The wavelet packet transform (WPT) is introduced as an alternative means of extracting time-frequency information from vibration signature. The resulting WPT coefficients provide one with arbitrary time-frequency resolution of a signal. With the aid of statistical-based feature selection criteria, many of the feature components containing little discriminant information could be discarded, resulting in a feature subset having a reduced number of parameters without compromising the classification performance. The extracted reduced dimensional feature vector is then used as input to a neural network classifier. This significantly reduces the long training time that is often associated with the neural network classifier and improves its generalization capability.**

Index Terms—**Condition monitoring, diagnosis, fault detection, wavelet transform.**

I. Introduction

Any major piece of industrial machinery equipment requires a certain degree of maintenance to assure successful operation over a long period of time. To achieve this objective, an automated condition monitoring system is needed. This health usage monitoring (HUM) system would allow early detection of potentially catastrophic faults that would be extremely expensive to repair. It also allows for implementation of condition based maintenance, and significant savings can be made by delaying scheduled maintenance until convenient or necessary. Generally, a simple condition monitoring system is approached from a pattern classification perspective. It can be decomposed into three general tasks: data acquisition, feature extraction, and condition classification [1]. The most common family of monitoring methods is based upon nondestructive vibration measurements using multiple accelerometers [2]–[8]. The general principle behind using vibration signals for monitoring involves those components in mechanical systems that vibrate during operation. When faults develop, some of the system dynamics vary, resulting in significant deviations in the vibration patterns. By employing appropriate data analysis algorithms, it is feasible to detect changes in vibration signatures caused by faulty components and to make decisions about the status of the machinery. In many of the classification systems currently used (i.e., neural-network-based systems, in particular), the process of feature extraction is inherently embedded in the classification technique rather than being identified as a separate process. If a multilayer neural network is used to classify unprocessed data, the input layer, which learns from examples, will essentially serve as a feature extractor. However, in problems such as vibration time-series data, the input dimensionality of the problem becomes an impediment to classification. Even neural networks are limited by the problem of parameter estimation—as the number of parameters increase, the amount of data required to train the neural network must increase to achieve satisfactory performance. For a complex problem, obtaining the necessary data may be expensive or even impossible. Feature extraction is needed to reduce the dimensionality of the data before performing classification. This is based upon the assumption that the important structure in the data actually lies in a much lower dimensional space. Feature extraction involves preliminary processing of sensor measurements to obtain suitable parameters that reveal whether an interesting pattern is emerging. It is generally not possible to classify machine conditions based upon an individual sample of the vibration. Therefore, a feature extraction technique is needed for preliminary processing of recorded time-series vibrations over a long period of time to obtain suitable parameters that, in linear and/or nonlinear combination, reveal whether a fault is evolving. In general, this requires windowing of the time-series vibration signals to form signal segments on which linear, bilinear, or nonlinear transformations are applied. The aim of feature extraction is to devise a transformation that extracts the signal features hidden in the original time domain. Corresponding to different characteristics of signals, transformations should be properly selected such that specific signal structure can be enhanced in its transformation domain. This would make the following decision-maker design (i.e., for fault classification) much easier.

Usually, the vibration signals of defective components are highly structured and can be grouped into two categories: sustained defects and intermittent defects [9]. For sustained defects, the signal is sinusoidal. Fourier-based analysis, which uses sinusoidal functions as basis functions, provides an ideal candidate for extraction of these narrow-band signals. For intermittent defects, features reflecting machinery faults in the pickup (windowed) time-series vibration signals neither appear in a repetitive manner nor consist of regular frequency components with

Manuscript received January 16, 1999; revised December 10, 1999. Abstract published on the Internet March 12, 2000. This work was supported in part by the U.S. Air Force Office of Scientific Research under Grant F49620-98-1-0049.

G. G. Yen is with the Intelligent Systems and Control Laboratory, School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078 USA (e-mail: gyen@ceat.okstate.edu).

K.-C. Lin was with the Intelligent Systems and Control Laboratory, School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078 USA. He is now with the IBM Server Development Group, Austin, TX 78753 USA (e-mail: kclin@us.ibm.com).

Publisher Item Identifier S 0278-0046(00)04753-5.

the evolution of time. Instead, these signals often demonstrate a nonstationary and transient nature, and carry small yet informative components embedded in larger repetitive signals. In this case, the Short Time Fourier Transform (STFT) can be employed to detect the localized transient. Unfortunately, the fixed windowing used in the STFT implies fixed time-frequency resolution in the time-frequency plane [10], [11]. The difficulty is that the accuracy of extracting frequency information is limited by the length of the window relative to the duration of the interesting signal. For example, in helicopter transmissions, important information concerning bearings can be on the order of tens of hundreds of hertz, whereas mesh frequencies and important fundamentals associated with gearing of the engine input, can be on the order of tens of thousands of hertz. To overcome the fixed time-frequency resolution problems, the recently developed wavelet based analysis [10], which provides flexible time–frequency resolution, becomes an efficient alternative in dealing with this type of machinery transient signals. Nonetheless, linear expansions in a single basis, whether Fourier or wavelet, are not flexible enough. The Fourier-basis analysis provides a poor representation of signals localized in time; while wavelet bases are not well adapted to represent signals whose Fourier transforms have narrow "high" frequency support because of poor resolution at high frequency. In both cases, it is difficult to detect and identify the signal pattern from the expansion coefficients because information is diluted across the whole basis. The wavelet packet transform (WPT) [12], on the other hand, uses a rich library of redundant bases with arbitrary time-frequency resolution. Therefore, it enables the extraction of features from signals that combine nonstationary and stationary characteristics.

The collection of all wavelet packet coefficients contains far too many elements to efficiently represent a signal. Care must be taken in choosing a subset of this collection in order to manage the computational complexity in practical situations. For classification applications, a natural direction is to address the issue of finding a wavelet-packet-based feature set that offers maximum feature separability due to class-specific characteristics. Our study explores the feasibility of the WPT as a tool in the search for features that may be used in the detection and classification of mechanical vibration signals. In particular, we formulate a systematic method of determining wavelet-packet-based features that exploit class-specific differences among interesting signals. This allows us to avoid human interaction. One could simply input a sample data set that represents the signals of interest and receive as output the dominant features that are suitable for classification purposes. In this study, we introduce a novel methodology for classifying vibration signals based on wavelet packet analysis. We suggest that such analysis can provide a more effective method of achieving robust classification over the more traditional single resolution techniques.

The study investigates the use of the wavelet-packet-based features in the classification of vibration signals. In Section II, we discuss the inefficiency of Fourier-based analysis for transient signal analysis and lead the reader to the wavelet-based analysis—wavelet transform and its generalization, the WPT. Section III presents an overview of the proposed classification system based on wavelet packet features. We first describe the feature measure, which will be used throughout. Then, we present two feature selection methodologies that aim to reduce the input dimension for the classifier. In Section IV, the feasibility of the proposed wavelet-packet-based feature extraction technique is demonstrated through numerical simulations of seed faults in the Westland transmission data set. We present our results and discuss the performance with respect to the parameters considered in our investigation. Finally, we conclude our study in Section V.

II. Time–Frequency Analysis of Vibration Signals

A. *Fourier-Based Analysis*

Vibration signal classification generally requires windowing of the time-series vibration signals to form signal segments on which linear, bilinear, or nonlinear transformations are applied. The Fourier based methods, in particular, the short-time Fourier transform (STFT), are usually employed for the extraction of narrow-band frequency content in signals. The difficulty with STFT is that the accuracy for extracting frequency information is limited by the length of this window relative to the duration of the signal. Specifically, the STFT of $x(t)$ is defined as

$$G(f,\tau) \equiv \int x(t) g^*(t-\tau) e^{-j2\pi f t}\, dt \tag{2.1}$$

where $g(t)$ is a window function. The STFT decomposes a signal in the time domain into a two-dimensional function in a time-frequency plane (f,τ). At a given frequency f, (2.1) is equivalent to filtering a signal at all times with a bandpass filter having as an impulse response the window function modulated to that frequency f. Alternatively, given a segment of signal windowed around time instant τ, one computes all frequencies of the STFT. Now, consider the ability of the STFT to discriminate between two pure sinusoids. Given a window function $g(t)$ and its Fourier transform $G(f)$, define the bandwidth Δf of the filter as

$$\Delta f^2 = \frac{\int f^2 |G(f)|^2\, df}{\int |G(f)|^2\, df}. \tag{2.2}$$

Then, two sinusoids will be discriminated only if they are more than Δf apart. Similarly, the spread in time is given by Δt defined as

$$\Delta t^2 = \frac{\int t^2 |g(t)|^2\, dt}{\int |g(t)|^2\, dt}. \tag{2.3}$$

So, two pulses in time can be discriminated only if they are more than Δt apart. Thus, the resolution in frequency of the STFT analysis is given by Δf, and the resolution in time is given by Δt. One important property, according to the uncertainty principle [13], is that for any suitably chosen window function, the time–bandwidth product of the window function has lower bound given by

$$\Delta t \Delta f = c \geq \frac{1}{4}\pi. \tag{2.4}$$

Here, c is a constant dependent on the choice of $g(t)$. Note that once the window function $g(t)$ is defined, the area (time–bandwidth product) of the window function in the time–frequency

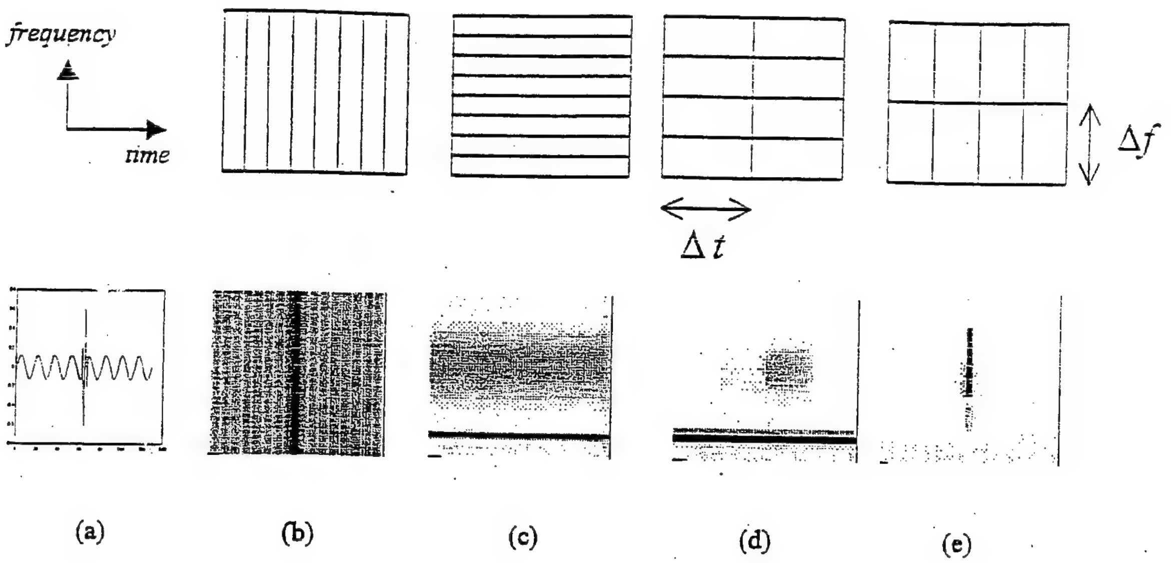


Fig. 1. Time-frequency representation of a signal using different length of analysis windows.

plane remains fixed. This means we cannot increase the time and frequency resolutions simultaneously. If we choose a window function with small Δt (good time resolution), then the corresponding frequency resolution will be poor (Δf will be large).

Consider analyzing a signal with different analysis window functions to demonstrate the possible deficiencies of fixed time-frequency resolution associated with the STFT. In Fig. 1, the signal $x(t)$, as shown in Fig. 1(a), contains a short-duration high-frequency component and a long-duration low-frequency component. Fig. 1(b)–(e) represents the different time-frequency representation corresponding to the specific analysis windows shown in the top row of Fig. 1. Fig. 1(b) corresponds to the time-domain representation. In Fig. 1(c), the frequency-domain representation, the burst is diluted across the broad frequency range, while the long-duration low-frequency component is effectively extracted. As we decrease the length of analysis window as illustrated in Fig. 1(d) and (e), the short-duration burst is more efficiently extracted. However, the long-duration low-frequency component becomes more ambiguous. Consequently, if we are analyzing the low-frequency content of a signal, we might desire a wide window function in time. On the contrary, if we were interested in high-frequency phenomena, a short-duration window function would be preferred. The STFT does not allow this flexibility, but, as we will see in the next section, wavelets give a framework for which this is automatic.

B. Wavelet Analysis

Fourier-based analysis is based on sinusoidal functions of various frequencies, whereas wavelet analysis, on the other hand, is founded on basis functions formed by dilation and translation of a prototype function $\psi(t)$, also known as a mother wavelet [14]. The wavelet basis function $\psi_{a,\tau}(t)$ is a family of short-duration high frequency and long-duration low-frequency functions defined as

$$\psi_{a,\tau}(t) \equiv \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-\tau}{a}\right), \quad a > 0, \quad \tau \in \Re. \tag{2.5}$$

The parameter τ indicates the translation in time, and the parameter a is the scale parameter. From the scaling property of Fourier transforms, if

$$\psi(t) \leftrightarrow \Psi(\Omega) \tag{2.6}$$

forms a Fourier transform pair, then

$$\frac{1}{\sqrt{a}}\psi\left(\frac{t}{a}\right) \leftrightarrow \sqrt{a}\Psi(a\Omega) \tag{2.7}$$

where $a > 0$ is a continuous variable. Thus, a contraction in one domain is accompanied by an expansion in the other, but in a nonuniform way over the time-frequency plane. Depending on the dilation parameter a, the wavelet function dilates or contracts in time causing the corresponding contraction or dilation in the frequency domain. When a is large ($a > 1$), the basis function becomes a stretched version of the mother wavelet ($a = 1$) and demonstrates a low-frequency characteristic. When a is small ($a < 1$), this basis function is a contracted version of the mother wavelet function and demonstrates a high-frequency characteristic.

Similar to the STFT, one can analyze a signal with continuous wavelet transform (CWT) which decomposes a signal in the time domain into a two-dimensional function in the *time–scale* plane (a, τ)

$$\Psi(a,\tau) \equiv \int x(t)\psi_{a,\tau}(t)\, dt. \tag{2.8}$$

The wavelet coefficient $\Psi(a,\tau)$ measures the *time–frequency* content in a signal indexed by the scale parameters and translation parameters. The term *frequency* instead of *scale* has been used in order to aid in understanding, since a wavelet with large-scale parameter is related to low-frequency content component, and vice versa. Thus, we see that the CWT corrects the noted deficiencies of the Fourier analysis as described in the previous section. That is, the CWT analyzes the low-frequency content of a signal with a wide duration function and, conversely, analyzes high-frequency phenomena with a short-duration function.

In practice, calculating wavelet coefficients at every possible scale using (2.8) is a fair amount of work and generates a lot of redundant data. It turns out that, if we limit the choice of a and τ in (2.5) to discrete numbers, then our analysis will be sufficiently accurate. In particular, if we choose scale and translation parameters (i.e., j and k, respectively) based on powers of two, then there exists $\psi(t)$ with good time-frequency localization property. The set of functions

$$\psi_{j,k}(t) \equiv 2^{j/2}\psi(2^j t - k), \quad j, k \in Z \tag{2.9}$$

constitute an orthonormal basis for $L^2(\Re)$ [15]. Here, Z denotes the set of integers, and $L^2(\Re)$ denotes the class of measurable functions $x(t)$ in $\Re$ satisfying:

$$\int_{\Re} |x(t)|^2\, dt < \infty. \tag{2.10}$$

Any signal $x(t)$ in $L^2(\Re)$ can then be expressed as

$$x(t) = \sum_{j,k} \langle f, \psi_{j,k}\rangle \psi_{j,k}(t). \tag{2.11}$$

This is called the discrete wavelet transform (DWT). In practice, the implementation of the DWT suitable for finite-length discrete-time signals is based upon the multiresolution analysis introduced by Mallat [16]. The development leads to a computationally efficient algorithm known as the fast wavelet transform (FWT). By introducing a new function, the scaling function $\phi(t)$, the orthogonal wavelets could be constructed and incorporated into a system that uses cascaded filters to decompose a signal. Specifically, the wavelet $\psi(t)$ is often generated from $\phi(t)$ through the following dilation equations:

$$\phi(t/2) = \sqrt{2}\sum_k h(k)\phi(t-k) \tag{2.12}$$

$$\psi(t/2) = \sqrt{2}\sum_k g(k)\phi(t-k). \tag{2.13}$$

Daubechies [15] has developed a procedure to solve the dilation equations such that the sequences $h(k)$ and $g(k)$ have only finite nonzero coefficients, which leads to a very efficient algorithm for computing wavelet coefficients. In general, $h(k)$ is the coefficient of the low-pass filter, whereas $g(k)$ represents a high-pass filter. This practical filtering algorithm is, in fact, a classical scheme known as a two-channel subband coding using quadrature mirror filters (QMF's) [17]. A consequence of multiresolution is that we can transform a signal into wavelets without using wavelets or scaling functions. In general, these functions do not exist as explicit functions; they are limits of iterations. To compute the wavelet transform, all we need are filters. Rather than taking the scalar product of the scaling function or the wavelet with the signal, we convolve the signal with these filters. Fig. 2 demonstrates the wavelet decomposition procedure and shows the time–frequency plane corresponding to a wavelet decomposition. In contrast with STFT, the time resolution becomes arbitrarily fine at high frequency, while the frequency resolution becomes arbitrarily fine at low frequencies. Note that, in FWT, the number of points is gradually decreased through successive decimation. Thus, if we start with a signal of 2^J points, then, in the following level, we have 2^{J-1} wavelet coefficients. Therefore, the maximum decomposition level is equal to J.

C. WPT

Whereas the wavelet transform provides one with more flexible time-frequency resolution properties as described, one possible drawback is that the frequency resolution is rather poor in the high-frequency region. Therefore, it faces some difficulties for discrimination between signals having close high-frequency components. Wavelet packets, a generalization of wavelet bases, are alternative bases that are formed by taking linear combinations of the usual wavelet functions [12], [18]. These bases inherit properties such as orthonormality and time–frequency localization from their corresponding wavelet functions. A wavelet packet function is a function with three indices: $W^n_{j,k}(t)$. As with usual wavelets, integers j and k are index scale and translation operations, respectively

$$W^n_{j,k}(t) = 2^{j/2}W^n(2^j t - k). \tag{2.14}$$

The index n is called *the modulation parameter* or the *oscillation parameter*. The first two wavelet packet functions are the usual scaling function and mother wavelet function, respectively

$$W^0_{0,0}(t) = \phi(t) \tag{2.15}$$

$$W^1_{0,0}(t) = \psi(t). \tag{2.16}$$

Wavelet packet functions for $n = 2, 3, \cdots$ are then defined by the following recursive relationships:

$$W^{2n}_{0,0}(t) = \sqrt{2}\sum_k h(k)W^n_{1,k}(2t-k) \tag{2.17}$$

$$W^{2n+1}_{0,0}(t) = \sqrt{2}\sum_k g(k)W^n_{1,k}(2t-k) \tag{2.18}$$

where $h(k)$ and $g(k)$ are the QMF associated with the predefined scaling function and mother wavelet function. To measure specific time-frequency information in a signal, we simply take the inner product of the signal and that particular basis function. The wavelet packet coefficients of a function f can be computed via

$$w_{j,n,k} = \langle f, W^n_{j,k}\rangle = \int f(t)W^n_{j,k}(t)\, dt. \tag{2.19}$$

The idea of the usual wavelet decomposition, as shown in Fig. 2, is generalized to describe the calculation of wavelet packet coefficients $w_{j,n,k}$ of a discrete-time signal. Computing the full wavelet packet decomposition (WPD) of a discrete-time signal involves applying both filters to the discrete-time signal

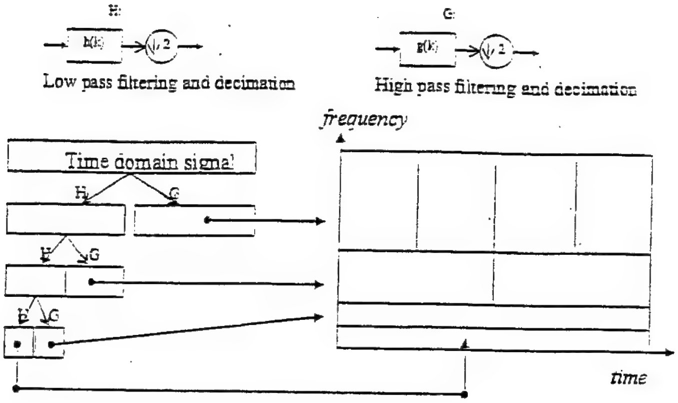

H:
H(k)
↓2
Low pass filtering and decimation
G:
g(k)
↓2
High pass filtering and decimation
frequency
Time domain signal
H
G
H
G
H
G
time

Fig. 2. Wavelet decomposition of time-domain signal.

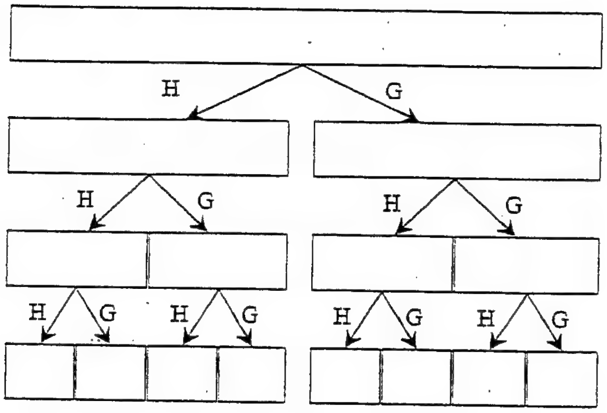


Fig. 3. Implementation of discrete WPD.

$[x_1, x_2, \cdots, x_n]$ and then recursively to each intermediate signal. The procedure is illustrated in Fig. 3.

Note that the method of decomposition described above does not result in a WPT tree displayed in increasing frequency order. This is because aliasing occurs, which exchanges the frequency ordering of some nodes of the tree. A simple swapping of the appropriate nodes results in the increasing frequency ordering referred to as the Paley ordering [18] of the tree, as shown in Fig. 4. Here, the dashed lines highlight the difference with Fig. 3. In this way, the leftmost node at each level will correspond to the lowest frequency band. In following sections, we will use this representation for easier interpretation.

Whereas the FWT decomposes only the low-frequency components, WPT decomposes the signal utilizing both the low-frequency components and the high-frequency components. This flexibility of a rich collection of abundant information with arbitrary time-frequency resolution allows extraction of features that combine nonstationary and stationary characteristics.

III. Feature Selection

A. Overview

The WPT is applied in classification problems based on time-series vibration signatures. First, the vibration data is decomposed via the WPT to extract the time–frequency-dependent information. Features are then defined based upon the WPD coefficients. Second, simple statistical processing based on discriminant analysis is applied to identify a set of robust features that provides the most discrimination among the classes of vibration data. Then, a neural network classifier is trained based on this reduced feature set. With statistical-based feature selection criteria, several feature components containing little discriminant information can be discarded, resulting in a feature subset having a reduced number of parameters. This will significantly ease the design of the neural classifier and enhance the generalization capability of the system. In the following sections, we define the WPD-based feature measurement used in this study. Then, we discuss some feature selection criteria and present the ones applied in this study to reduce the number of feature variables.

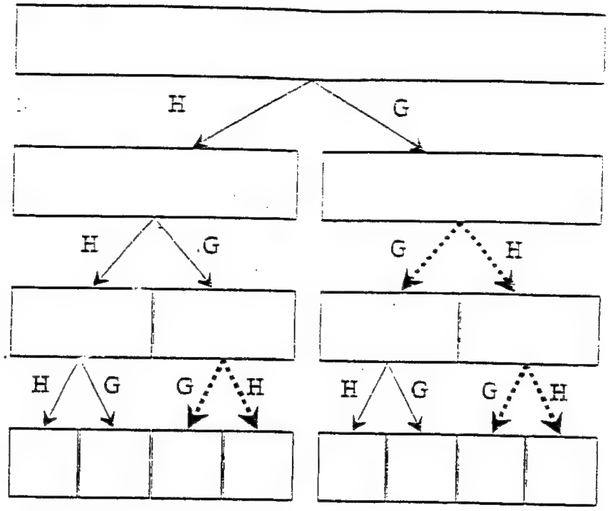


Fig. 4. The WPD tree displayed in Paley order.

B. Feature Measure

One deficiency that wavelet bases inherently possess is the lack of a translation-invariant property. To illustrate this by example, consider two signals with a slight shift in time, as shown in the left-hand side of Fig. 5. When the two signals are decomposed via the WPT, we can see appreciable differences between the two representations of the signals as shown in the right-hand side of Fig. 5 (a darker color corresponds to a larger WPD coefficient value). Therefore, direct assessment from all wavelet packet coefficients often turns out to be tedious or leads to inaccurate decisions.

Recall that each wavelet packet coefficient is given by

$$w_{j,n,k} = \langle f, W_{j,k}^n \rangle = \langle f, 2^{j/2} W^n(2^j t - k) \rangle \qquad (3.1)$$

where j is a scaling parameter, k is a translation parameter, and n is an oscillation parameter. Each $w_{j,n,k}$ coefficient measures a specific subband frequency content, controlled by the scale parameter j and the oscillation parameter n, of a signal around time instant $2^j t$.

We define the wavelet packet node energy as

$$e_{j,n} \equiv \sum_k w_{j,n,k}^2 \qquad (3.2)$$

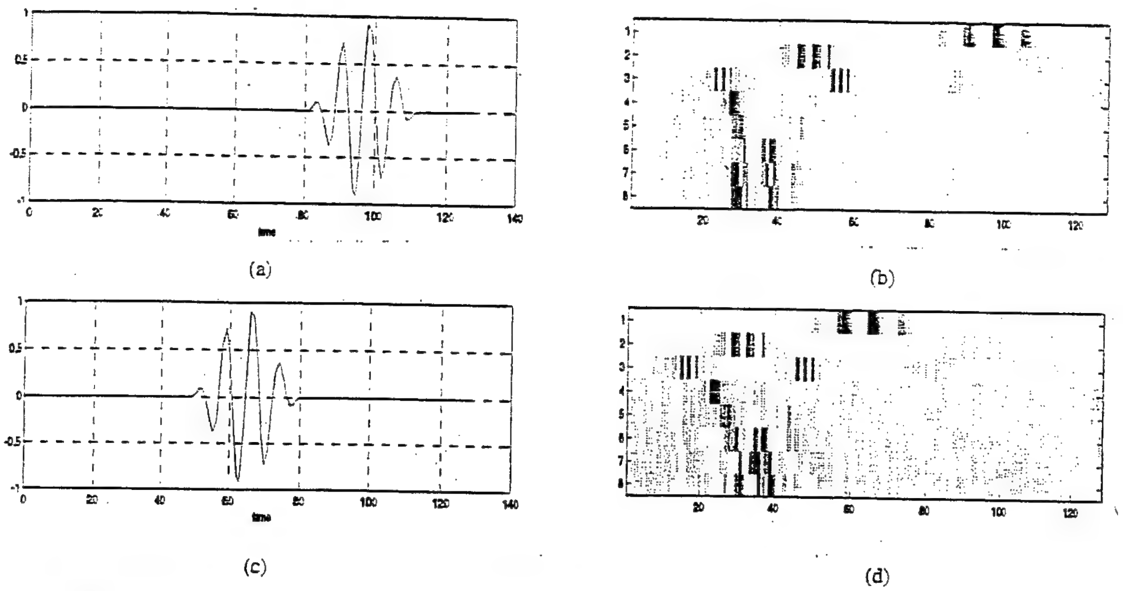


Fig. 5. WPD of time-shifted signals. (a) s1. (b) WPT of s1. (c) s2. (d) WPT of s2.

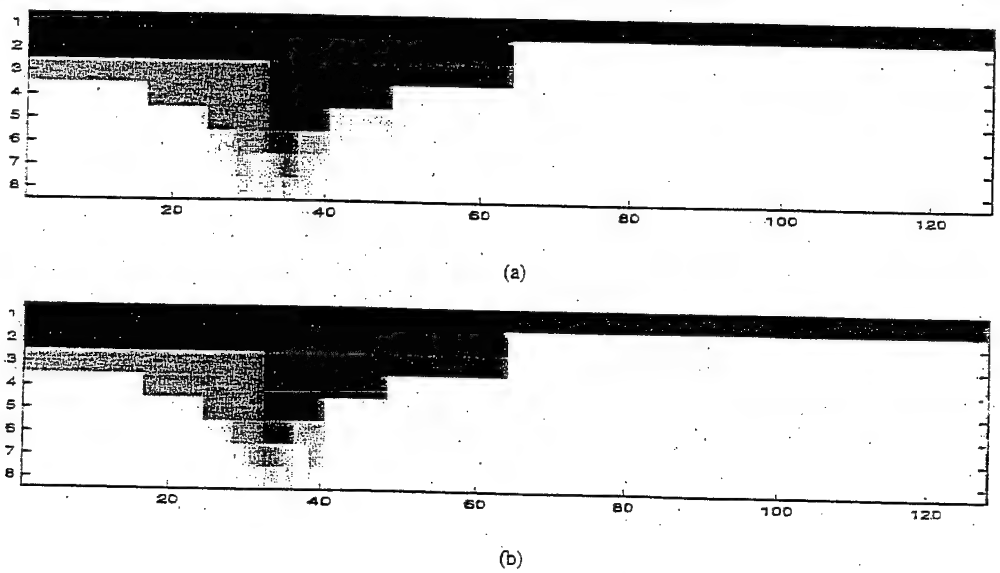


Fig. 6. Wavelet packet node energy of time-shifted signals. (a) Energy map of s1. (b) Energy map of s2.

which measures the signal energy contained in some specific frequency band indexed by parameters j and n. In the following, we will call each (j, n) a wavelet packet node. Fig. 6 displays the energy distribution that is calculated based on all coefficients in each wavelet packet node of the two signals given in Fig. 5. We can see those node energy values at level two, three, or four show no clear difference between the two signals. This example reveals that the node energy representation provides us with a more robust signal feature for classification than using coefficients directly. In our strategy, each wavelet packet node energy value was defined as an individual feature component and was used as a robust rudimentary exploration of the specific signal features that provide useful information for classification purposes.

C. Feature Dimension Reduction

One advantage of using WPT to decompose a signal is that it allows us to examine different time-frequency resolution components in a signal. For example, by computing the full WPD on a signal segment with $n = 2^J$ points for r resolution levels (where J and r are positive integers), the result is a group of $2^1 + 2^2 + \cdots + 2^r = 2^{r+1} - 2$ sets of coefficients where each set corresponds to a wavelet packet node. If the node energy as described before is used as a feature, we can obtain $2^{r+1} - 2$ feature components. However, direct manipulation of a whole set of node energies is prohibitive because the space normally has very high dimensionality, and the existence of undesired components makes the classification unnecessarily difficult. In the training of a neural network classifier, it is desirable to use a lower dimensional vector as input to the neural network to ease

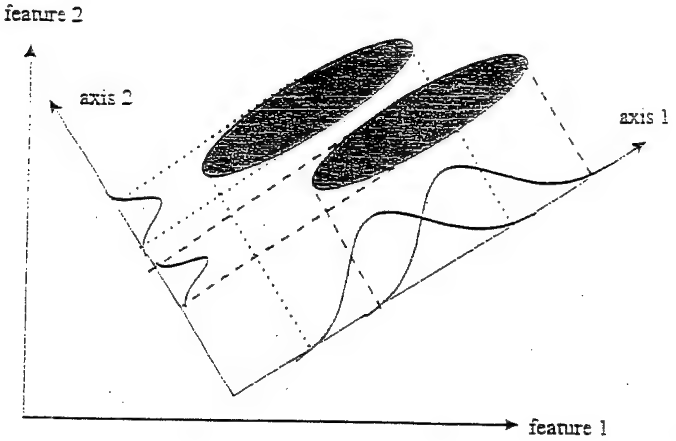


Fig. 7. Example of feature extraction for classification.

the design of the classifier and improve its generalization capability.

One popular technique in reducing the feature dimensionality is the Karhumen–Loéve (K–L) transform [19]. The K–L transform is optimal for "*signal representation*" in the sense that it provides the smallest mean-square error for a given number of data. However, the features defined by the K–L transform are not optimal for "*class separability*." As an example, the data from two-class categories with a Gaussian distribution are shown in Fig. 7. In the sense of K–L transform, the principal axis 1 with a larger eigenvalue is a better vector than axis 2 to represent the vectors of this distribution. That is, the selection of axis 1 produces a smaller mean-square error of representation than the selection of axis 2 alone. However, as seen in Fig. 7, if the two distributions are mapped onto axis 1, the marginal density functions are heavily overlapped. On the other hand, if they are mapped onto axis 2, the marginal densities are well separated. Therefore, for classification purposes, axis 2 is a better feature than axis 1 alone, preserving more classification information.

As described previously, it is not the mean-square error, as in the sense of K–L transform, but the classification accuracy that should be considered a primary criterion for reducing the feature dimension. The ability to classify patterns relies on the implied assumption that different classes occupy distinct regions in the pattern space. Intuitively, the more distant the classes are from each other, the better the chance of successful recognition of class membership of patterns. One transformation associated with this assumption is based on within- and between-class scatter matrices that are used in linear discriminant analysis (LDA) of statistics [20]. The idea is to find a linear transformation that projects the samples onto a lower dimensional space in which the variability of samples within each class is as close as possible, and the dispersion of the class mean vectors about the mean vector is as separated as possible.

Specifically, consider an L-class problem. The class sample covariance matrices measure the variability of samples within each class

$$S_c = (1/N_c) \sum_{i=1}^{N_c} (x_i^c - m_c)(x_i^c - m_c)^T, \qquad c = 1, 2, \cdots, L \quad (3.3)$$

where x_i^c is a sample vector belonging to class c, N_c is the number of samples belonging to class c, and m_c is the mean vector of class c

$$m_c = (1/N_c) \sum_{i=1}^{N_c} x_i^c. \quad (3.4)$$

As a result, the overall within-class variability can be estimated by the sample covariance matrix

$$S_w = \sum_{c=1}^{L} p_c S_c \quad (3.5)$$

where p_c is the *a priori* probability of class c. Similarly, the between-class covariance matrix measures the dispersion of the class mean vectors about the overall mean vectors

$$S_b = \sum_{c=1}^{L} p_c (m_c - m)(m_c - m)^T \quad (3.6)$$

where m represents the expected vector of the mixture distribution and is given by

$$m = \sum_{c=1}^{L} p_c m_c. \quad (3.7)$$

Now, if $\bar{x} = A^T x$ denotes a linear transformation of the original variables, then the between- and within-class covariance matrices in the transformed space can be found as $\overline{S_b} = A^T S_b A$ and $\overline{S_w} = A^T S_w A$. The goal is to find a subspace where the ratio of S_b and S_w is maximized. In this case, it may be measured by the ratio of the determinant of the preceding matrices (i.e., the determinant, being the product of the eigenvalues, is the product of the variance in the principal directions). The problem could, thus, be formulated so as to find a transformation $\bar{A}$ such that

$$\bar{A} = \arg\max_{A} \frac{|A^T S_b A|}{|A^T S_w A|}. \quad (3.8)$$

The solution for (3.8) is given by the $\min(n, L-1)$ eigenvectors of $S_w^{-1} S_b$ where n is the dimension of the original data set [20]. Once the transformation map $\bar{A}$ is obtained, then the feature vector $\bar{A}^T x$ is computed for each sample and, finally, it is assigned to the class which has the mean vector closest to this feature vector. Although the vector found by LDA works well in most cases, several drawbacks might occur in practice. First, when we apply LDA to extract the discriminant feature vector, the mathematical procedure automatically combines the feature extractor and the classifier in a linear form. By restricting the form or criterion of the mapping, we implicitly assume an over-simplistic model of the pattern recognition system. Such a situation will arise if the classes are not linearly separable and we restrict the feature extractor to a linear form.

Moreover, since LDA involves the computation of the inverse of the covariance matrices, it may lead to numerical problems, especially when the matrices are estimated based on a limited data set. In our application on the classification of vibration signal data collected from multisensors, we might have thousands of time-frequency feature components, while only hundreds of training samples are available. For example, given a 256-point signal, full decomposition of the signal to the seventh level and use of node energy as the feature component will result in a 254-dimensional feature vector. Combining all feature vectors from multiple sensors, say eight, will result in a 2032–dimensional vector. However, only a few eigenvalues, such as ten, are dominant, i.e.,

$$\lambda_1 + \lambda_2 + \cdots + \lambda_{2032} \cong \lambda_1 + \cdots + \lambda_{10}. \tag{3.9}$$

This means that, in a practical sense, we are handling S_w with rank ten, even though the mathematical rank of S_w is still 2032, i.e., $\lambda_i \neq 0, \forall i$. In the calculation of S_w^{-1}, the determinant of S_w is $\prod_{i=1}^{2032} \lambda_i$ and $\lambda_i, i = 11, \cdots, 2032$ are very close to zero. Suppose $\lambda_1 + \cdots + \lambda_{10} = 0.9$ out of $\sum_{i=1}^{2032} \lambda_i = 1$, then

$$\prod_{i=1}^{10} \lambda_i \times \prod_{j=11}^{2032} \lambda_j = \prod_{i=1}^{10} \lambda_i \times (0.1/2022)^{2022} \cong 0 \tag{3.10}$$

for the assumption $\lambda_{11} = \lambda_{12} = \cdots = \lambda_{2032} = 0.1/2022$.

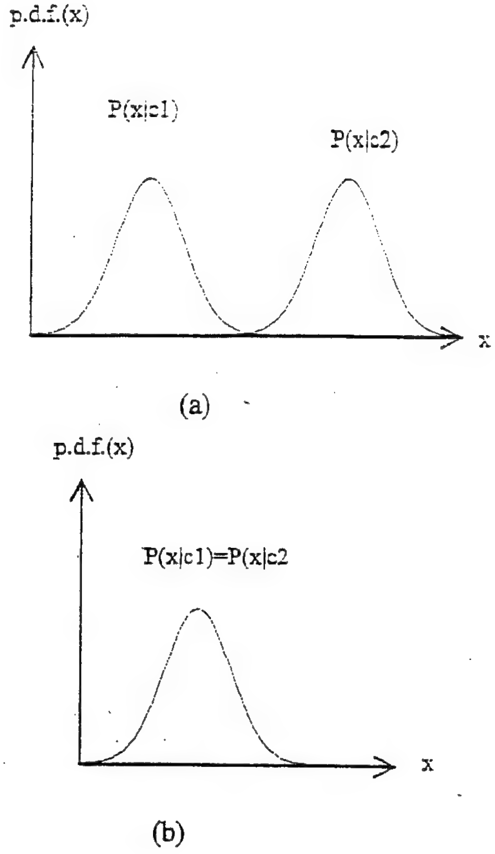
This, indeed, leads to some numerical instability in handling such a near-singular matrix. For this reason, we resort to employing the feature selection in feature measurement as described in the following section, which considers the numerical problems of calculating the inverse of covariance matrices as LDA does. Instead of trying to find a linear transformation to reduce the dimensionality, we evaluate the discriminant power of each individual feature component and discard those feature components containing little class separability information as measured by selected criterion. Then, a neural network is employed as a classifier to deal with nonlinearly separable cases in the feature space.

The idea behind feature selection in feature measurement space is to select the feature components that contain discriminant information and discard those feature components that provide little information useful for classification purposes [21]. Specifically, the feature component $\{f_k \mid k = 1, 2, \cdots, n\}$ is ranked

$$J(f_1) \geq J(f_2) \geq \cdots \geq J(f_d) \geq \cdots \geq J(f_n) \tag{3.11}$$

where $J(\cdot)$ is a criterion function for measuring the *discriminant power* of a specific feature component. The feature subset can be selected from the available features that have larger criterion function values.

To obtain a clearer picture of measuring the discriminant power of a feature, it is essential to introduce the concept of probabilistic structure of classes. Consider the probability density function of class $c1$ and $c2$ given in Fig. 8. For a specific feature variable x, if $p(x \mid c1)$ is zero for all x such that $p(x \mid c2) \neq 0$ as illustrated in Fig. 8(a), then these two classes can be fully separable. On the other hand, when $p(x \mid c1) = p(x \mid c2)$ as in Fig. 8(b), it is impossible to distinguish elements of class $c1$ from those belonging to $c2$. Intuitively, a criterion function for evaluating the discriminant power of a feature could be assessed by measuring the overlap between $p(x \mid c1)$ and $p(x \mid c2)$. A high overlap corresponds to a low discriminant power and vice versa.



(a)

(b)

Fig. 8. Probability density functions of (a) two well-separated classes and (b) two completely overlapping classes.

TABLE I
WESTLAND HELICOPTER GEARBOX DATA DESCRIPTION

Fault Type Number	Description
1	No Defect
2	Planetary Bearing Corrosion
3	Input Pinion Bearing Corrosion
4	Spiral Bevel Input Pinion Spalling
5	Helical Input Pinion Chipping
6	Helical Idler Gear Crack Propagation
7	Collector Gear Crack Propagation
8	Quill Shaft Crack Propagation

In general, a criterion function for measuring the overlap between classes has the following properties [21].

1) The measure is minimum when the conditional probability density function for class $c1$ and $c2$ are identical, i.e.

$$J(\cdot) = 0, \quad \text{if } p(x \mid c1) = p(x \mid c2). \tag{3.12}$$

2) The measure is nonnegative.
3) The measure attains a maximum when the classes are disjoint, i.e.,

$$J(\cdot) = \max, \qquad \text{if } p(x \mid c1) = 0 \ \text{ when } p(x \mid c2) \neq 0, \quad \forall x. \tag{3.13}$$

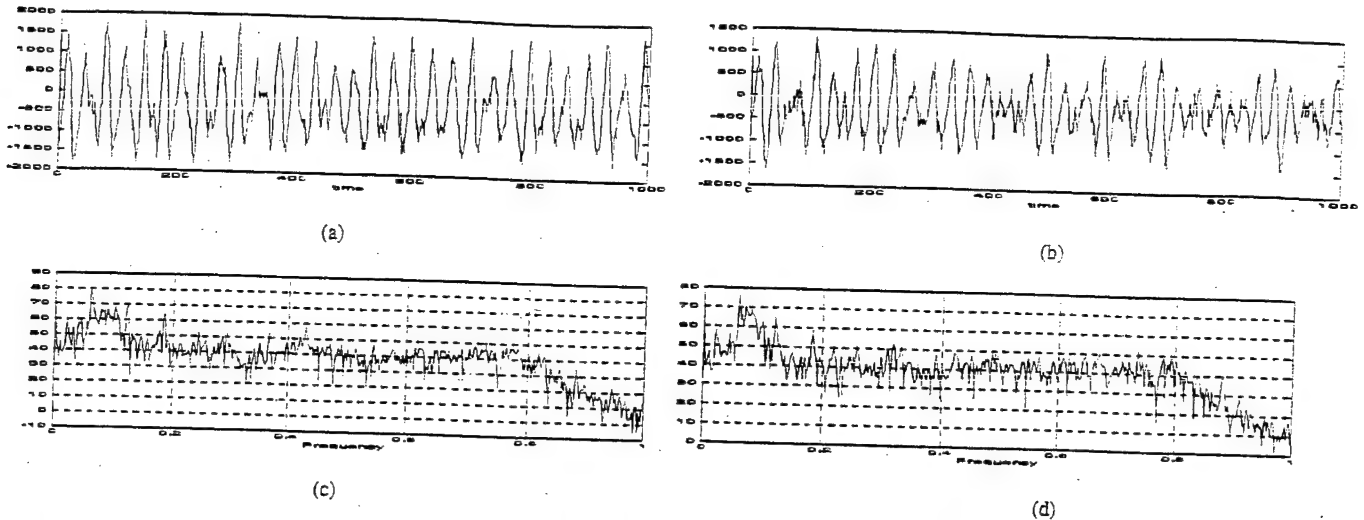


Fig. 9. Typical vibration signals and corresponding power spectral density (PSD) for normal mode and fault 3. (a) Normal mode. (b) Fault 3. (c) PSD of normal mode. (d) PSD of fault 3.

TABLE II
DIMENSION OF FINAL FEATURE VECTOR USING ONE SENSOR

	Wavelet packet feature							
	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6	Sensor 7	Sensor 8
PWM	32	33	42	31	38	47	39	50
KNK	34	36	46	45	34	34	50	33
	Fourier Feature							
	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Sensor 6	Sensor 7	Sensor 8
PWM	31	37	31	31	43	51	37	54
KNK	28	30	37	35	31	35	37	35

TABLE III
CLASSIFICATION PERFORMANCE (SENSORS 1 AND 2).

		Sensor 1		Sensor 2	
		WPT	FT	WPT	FT
PWM	Tr. Err.	3.25	0.75	1	0
	Test Err.	21.92	4.25	4	2.17
KNK	Tr. Err.	2.75	1.25	1.00	0.25
	Test Err.	24.50	4.58	4.33	1.92

TABLE IV
CLASSIFICATION PERFORMANCE (SENSORS 3 AND 4)

		Sensor 3		Sensor 4	
		WPT	FT	WPT	FT
PWM	Tr. Err.	0.75	0.25	2.25	1.25
	Test Err.	6.75	1.75	6.42	5.83
KNK	Tr. Err.	0.75	0.50	1.25	1.75
	Test Err.	7.25	1.42	8.00	5.08

Although the above properties provide an intuitive justification of their suitability for feature selection, their relative potential can be assessed only if their relationship to the classification error is known. Nevertheless, these measures are closely related to the error probability. This relationship is a consequence of the fact that the measures give a direct indication of the amount of the overlap of the class probability densities. In this study, however, we adopt a simple yet efficient criterion function known as Fisher's criterion [20]. In a two-classes problem it is given by

$$J_{f_k}(i,j) = \frac{|\mu_{i,f_k} - \mu_{j,f_k}|^2}{\sigma^2_{i,f_k} + \sigma^2_{j,f_k}} \tag{3.14}$$

where μ_{i,f_k}, μ_{j,f_k} are the mean values of the kth feature, f_k, for class i and j, and σ^2_{i,f_k}, σ^2_{j,f_k} are the variance of

TABLE V
CLASSIFICATION PERFORMANCE (SENSORS 5 AND 6)

		Sensor 5		Sensor 6	
		WPT	FT	WPT	FT
PWM	Tr. Err.	0.25	0.75	0.75	1.25
	Test Err.	50.33	50.33	62.92	57.92
KNK	Tr. Err.	1.25	1.25	4.00	3.25
	Test Err.	50.83	53.67	61.58	59.75

TABLE VI
CLASSIFICATION PERFORMANCE (SENSORS 7 AND 8)

		Sensor 7		Sensor 8	
		WPT	FT	WPT	FT
PWM	Tr. Err.	1	1.5	0.5	0
	Test Err.	2.25	2.42	62	46.83
KNK	Tr. Err.	1.50	0.00	3.25	1.75
	Test Err.	5.08	3.08	61.08	45.17

the kth feature, f_k, for class i and j correspondingly. When there are more than two classes of data, the general approach is to take the summation of the pairwise combinations of $J_{f_k}(i,j)$

$$J_{f_k} = \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} J_{f_k}(i,j) \tag{3.15}$$

as an estimation of discriminant power for the specific feature f_k. Here, L represents the number of classes in the problem. Equation (3.15) provides us with a measure to evaluate the effectiveness of the "*global*" feature that is simultaneously suitable to differentiate all classes of signals. For a small number of classes, this approach may be sufficient. The more signal classes, the more ambiguous (3.15) becomes. A large value for (3.15) may be due to a few significant terms with negligible majority (a favorable case) or to the accumulation of many terms with relatively small values (an unfavorable case). A feature that can effectively differentiate a pair of signal classes, i.e., with a large discriminant measure as calculated by (3.15), might be averaged during the pairwise summation. Note the feature selection is performed to ease the neural network classifier design (i.e., in training process). To avoid such a problem, we propose two feasible approaches as described below.

1) Approach I (PWM): Instead of trying to identify features that are effective for the entire multiclass problem globally as measured by (3.15), we select a feature subset based on (3.14) for each possible pair of classes. Then, we take the union of feature components selected from each pair of classes to form the final feature vector. Specifically, given an L-class problem with n feature components, the selection process is detailed in the following.

1) For each possible class pair $\{(i,j)\,|\,i = 1,2,\cdots,L-1,\ j = i+1, i+2,\cdots,L\}$, calculate the discriminant power measure for each feature component, f_k, i.e.,

$$J_{f_k}(i,j) = \frac{|\mu_{i,f_k} - \mu_{j,f_k}|^2}{\sigma_{i,f_k}^2 + \sigma_{j,f_k}^2}. \tag{3.16}$$

2) For each class pair (i,j), sort $J_{f_k}(i,j)$ such that

$$J_{f_1}(i,j) \geq J_{f_2}(i,j) \geq \cdots \geq J_{f_d}(i,j) \geq \cdots \geq J_{f_n}(i,j). \tag{3.17}$$

Determine the feature subset $F_{i,j}$ for each class pair by selecting d feature components that have maximum $J_{f_k}(i,j)$ value

$$F_{i,j} = \{f_k \,|\, k = 1,2,\cdots,d\},$$
$$i = 1,2,\cdots,L-1; \quad j = i+1, i+2,\cdots,L. \tag{3.18}$$

3) Form the final feature set by taking the union of each feature subset

$$F_{\text{final}} = \left\{ \bigcup_{i=1}^{L-1} \bigcup_{j=i+1}^{L} F_{i,j} \right\}. \tag{3.19}$$

2) Approach II (KNK): Another approach to avoid the influence of the pairwise summation process is similarly suggested by Watanabe [22]. Given an L-class signal classification problem, we can consider the class q signals as the conceptual opposite of the class $\bar{q}$ signals, which is the ensemble of data belonging to classes other than the q class. Then, we apply Fisher's criterion as was done in the two-class problems to evaluate the discriminant power of each individual feature component.

1) For each class $q = 1,2,\cdots,L$, we partition the data set to class q signals and class $\bar{q}$ signals. In this way, we can get L sets of data that can be used for selecting features.
2) For each of the L sets, use Fisher's criterion to evaluate the discriminant power for each feature component

$$J_{f_k}(q,\bar{q}) = \frac{|\mu_{q,f_k} - \mu_{\bar{q},f_k}|^2}{\sigma_{q,f_k}^2 + \sigma_{\bar{q},f_k}^2}. \tag{3.20}$$

3) For each of the L sets, sort $J_{f_k}(q,\bar{q})$ such that

$$J_{f_1}(q,\bar{q}) \geq J_{f_2}(q,\bar{q}) \geq \cdots \geq J_{f_d}(q,\bar{q}) \geq \cdots \geq J_{f_n}(q,\bar{q}). \tag{3.21}$$

TABLE VII
DIMENSION OF FINAL FEATURE VECTOR USING EIGHT SENSORS

	Waveiet packet feature							
	d=1	d=2	d=3	d=4	d=5	d=6	d=7	d=8
PWM	7	14	26	35	42	50	58	62
KNK	8	16	24	32	39	47	55	63
	Fourier Feature							
	d=1	d=2	d=3	d=4	d=5	d=6	d=7	d=8
PWM	9	16	24	27	29	30	32	38
KNK	8	16	24	32	40	48	51	56

TABLE VIII
CLASSIFICATION PERFORMANCE (EIGHT-SENSOR DATA; PWM)

		d=1	d=2	d=3	d=4	d=5	d=6	d=7	d=8
WPT	Tr. Err.	0.5	0	0	0	0	0	0	0
	Test Err.	0.92	0.17	0	0.17	0.25	0	0.08	0.25
FT	Tr. Err.	0	0	0	0	0	0	0	0
	Test Err.	0	0	0	0.25	0	0	0	0

TABLE IX
CLASSIFICATION PERFORMANCE (EIGHT-SENSOR DATA; KNK)

		d=1	d=2	d=3	d=4	d=5	d=6	d=7	d=8
WPT	Tr. Err.	0	0.25	0	0	0	0	0	0
	Test Err.	0.08	0	0	0	0	0.08	0.17	0.08
FT	Tr. Err.	2.75	0	0	0	0	0	0	0
	Test Err.	2.17	0.33	0	0.08	0	0	0	0

Determine the feature subset $F_{q,\bar{q}}$ for each of L set by selecting d feature components that have maximum $J_{f_k}(q,\bar{q})$ value

$$F_{q,\bar{q}} = \{f_k \mid k = 1, 2, \cdots, d\}, \quad q = 1, 2, \cdots, L. \tag{3.22}$$

4) Form the final feature set by taking the union of each feature subset

$$F_{\text{final}} = \left\{ \bigcup_{q=1}^{L} F_{q,\bar{q}} \right\}. \tag{3.23}$$

Suitable feature components, which offer favorable class separability measure, are found as described. Many classifiers can then be designed based on these features. A feedforward neural network is employed in this study because of its capability in dealing with nonlinearly separable distributions.

D. Neural Networks Classifier

Once suitable features have been extracted and selected from the vibration data as discussed above, it is necessary to determine the failure mode based upon these features. Ideally, the features for normal and faulty conditions will occupy nonoverlapping areas in the feature space. If not, then the classification algorithm will have to approximate a Bayes classifier [23].

Consider a L-class problem. The probability that a particular pattern x, comes from class $c_i, i = 1, 2, \cdots, L$ is denoted as $p(c_i \mid x)$. If the pattern classifier decides that x came from c_j when it actually came from c_i, it incurs a loss, denoted $l(i \mid j)$. As pattern x may belong to any one of L classes under consideration, the average loss incurred in assigning x to class c_j is

$$r_j(x) = \sum_{k=1}^{L} l(k \mid j) p(c_k \mid x). \tag{3.24}$$

In general, the loss for a correct decision is zero (i.e., $k = j$) and it has the same nonzero value (e.g., 1) for any incorrect decision, i.e.,

$$l(k \mid j) = 1 - \delta_{k,j} \tag{3.25}$$

where

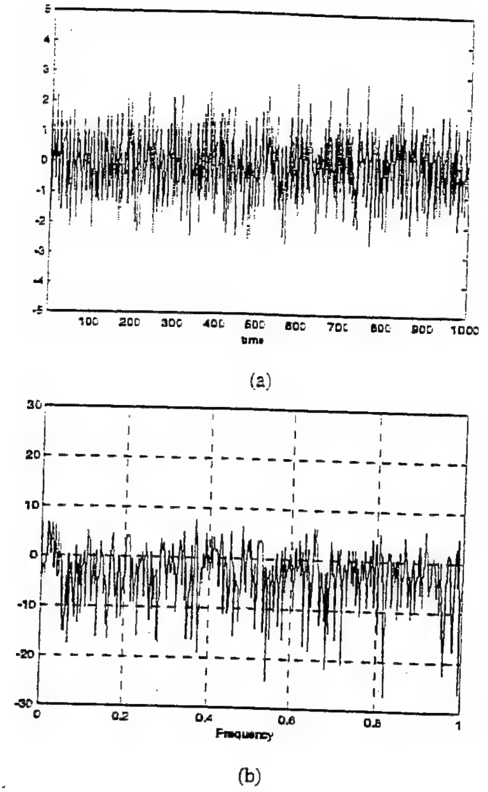
$$\delta_{k,j} = \begin{cases} 1, & k = j \\ 0, & k \neq j. \end{cases} \tag{3.26}$$

Then, the loss of assigning a pattern x to class c_j becomes

$$r_j(x) = \sum_{k \neq j} p(c_k \mid x). \tag{3.27}$$

The classifier has L possible classes to choose from for any given unknown pattern x. If it computes $r_j(x), j = 1, 2, \cdots, L$, for each pattern x, and assigns the pattern to the class with smallest loss, then total average loss with respect to all decisions will be minimum. The classifier that minimizes (3.27) is called the *Bayes Classifier*. Thus, the Bayes classifier assigns an unknown pattern vector x to class c_i if

$$r_i(x) < r_j(x), \quad j = 1, 2, \cdots, L; \quad j \neq i. \tag{3.28}$$



(a)

(b)

Fig. 10. White Gaussian noise and its power spectrum. (a) White noise. (b) PSD of white noise.

Substituting (3.27) into (3.28), the decision rule is then to choose label c_i if

$$\sum_{k\neq i} p(c_k \mid x) < \sum_{k\neq j} p(c_k \mid x), \qquad j = 1, 2, \cdots, L; \quad j \neq i. \quad (3.29)$$

Note that each side of (3.29) has all but one term missing. The decision rule then becomes to assign x to c_i if, for all $j = 1, 2, \cdots, L, j \neq i$

$$p(c_i \mid x) > p(c_j \mid x). \quad (3.30)$$

For the decision rule based on (3.27) to hold, the *posteriori* density functions $p(c_i \mid x); i = 1, 2, \cdots, L$ must be known. In practice it must be estimated from the available data set. To obtain the estimates of the posteriori density functions, neural networks are applied in the study for the following reasons. First, neural networks are universal approximators in the sense that they can theoretically approximate any continuous input-output mapping to any desired degree of accuracy. Hence, they can be used to approximate the *posteriori* function $p(c_i \mid x)$. Secondly, neural networks are inherently nonlinear in the activation function; they have the ability to capture the underlying nonlinearity for the generation of incoming data.

IV. Simulation Results

A. Data Description

In this section, the feasibility of the wavelet-packet-based feature classification technique was examined through numerical simulations on a real data set known as the Westland data set. The Westland data set [24] was chosen because it has been analyzed by a number of other researchers and because it is considered a benchmark data set in the field. The vibration data used for simulation is archived at the Applied Research Laboratory, Pennsylvania State University, University Park. In this data set, vibration data was recorded from an aft main power transmission of a U.S. Navy CH-46E helicopter. The vibration data were collected using eight accelerometers mounted at the known fault sensitive locations of the helicopter gearbox. The data were then recorded for various seeded faults including the no-defect case, listed in Table I. Nine torque levels, ranging from 27% up to 100% and various fault severity levels were applied. One tachometer was placed on the aft transmission in place of the rotor position motor. The tach signal is a 256 pulse-per-revolution signal with a once-per-revolution signal superimposed on it. Based on its position in the gearbox, one revolution describes a complete rotation of the rotor position output, not that of the main shaft. The vibration data was sampled at 103 116.08-Hz rate. With the approximate 100-kHz sampling rate, there are between 897–904 samples within the period defined by the tachometer signal.

B. Signal Segmentation

For utilization of the fast WPT algorithm, each of the 1024 time-series data points of vibration signal is defined as a sample vector to be analyzed. The reason for using 1024 points is it covers one period defined by the tachometer period. It is reasonable to assume that fault symptoms can be fully described within the period. Fig. 9 shows two signal segments and corresponding power spectra for normal mode and fault 3. Looking at the spectrum of the vibration data segment, we observe a long flat region toward the end of the frequency range; it is, thus, inferred that the bandwidth of the signal is much less than the sampling frequency. Based on this observation, the sample vector is first downsampled by four to yield a 256-point signal segment. This lowers the computational complexity without losing much information of the original signal.

Additionally, if a random signal has a nonzero mean, its power spectrum has an impulse at zero frequency. If the mean is relatively large, this component will dominate the spectrum estimate, causing low-amplitude low-frequency components to be obscured by the leakage. Therefore, in practice, the mean is often estimated, and the resulting estimate is subtracted from the random signal before computing the power spectrum estimate. Although the sample mean is only an approximate estimate of the zero-frequency components, subtracting it from the signal often leads to a better estimate at neighboring frequencies [25].

C. Generation of Training Data Set/Testing Data Set

There are total of 68 data sets available, which correspond to nine different torque levels and eight-class conditions. For each torque level, not all fault signals are available. Each file contains 412 464 data points. The 412 464 data points are segmented to 400 sample segments containing 1024 data points each. In this study, the first 50 samples collected represent the training data set, while the remaining 350 samples are used as the testing data set. These unseen data sets to the neural classifier are kept to evaluate the generalization capability. In this study, only the data

TABLE X
CLASSIFICATION PERFORMANCE (WHITE NOISE; SNR = 0 dB; PWM)

		d=1	d=2	d=3	d=4	d=5	d=6	d=7	d=8
WPT	Tr. Err.	0.25	0	0	0	0	0	0	0
	Test Err.	0.5	0.08	0.17	0.08	0.08	0.33	0.33	0.42
FT	Tr. Err.	14.25	1.25	1	0.25	0.5	0	0	0
	Test Err.	17.83	2.25	2.5	2.83	2.08	2.17	1.25	1.75

TABLE XI
CLASSIFICATION PERFORMANCE (WHITE NOISE; SNR = 0 dB; KNK)

		d=1	d=2	d=3	d=4	d=5	d=6	d=7	d=8
WPT	Tr. Err.	0	0.25	0.25	0	0	0	0	0
	Test Err.	0.25	0.5	1.08	0.08	0.17	0.17	0.25	0.42
FT	Tr. Err.	1.75	1	0.25	1	0.5	0	0	0.25
	Test Err.	6.33	6.08	3.42	2.08	4	2.25	2.17	1.5

TABLE XII
CLASSIFICATION PERFORMANCE (WHITE NOISE; SNR = −3 dB; PWM)

		d=1	d=2	d=3	d=4	d=5	d=6	d=7	d=8
WPT	Tr. Err.	0.25	0	0.25	0.25	0	0.25	0	0
	Test Err.	0.67	0.08	0.92	0.83	0	0.17	0.08	0.08
FT	Tr. Err.	5	1	1.25	1.25	0.75	1	0.25	0
	Test Err.	7.25	5.42	5.92	6.92	4.83	4.83	6.67	5

TABLE XIII
CLASSIFICATION PERFORMANCE (WHITE NOISE; SNR = −3 dB; KNK)

		d=1	d=2	d=3	d=4	d=5	d=6	d=7	d=8
WPT	Tr. Err.	0.5	0	0	0	0.25	0	0.25	0.25
	Test Err.	1	0.17	0.17	0.33	0.75	0.08	1.17	1.33
FT	Tr. Err.	16.25	3.25	1.5	1.75	1	0.75	0.25	0
	Test Err.	19	10.58	5	6	4.5	5.08	4.83	4.17

sets corresponding to torque level 100% were used for evaluation.

D. System Description

In the following simulations, each vibration signal segment is transformed into a wavelet-packet-based energy vector as described in Section III-B. The proposed two-feature selection method is then employed to identify a subset of feature components that will be used as input to the neural network classifier. The steps are summarized as follows.

1) Seven-level WPD is generated for each vibration signal segment.
2) Energy-based feature measures, as discussed in Section III-B, are then computed for each WPD of signal segment from step 1. This results in a 254-dimensional feature vector.
3) Identify a subset of feature components, as discussed in Section III-C, to form an input vector for the neural network classifier.

E. Test Result Using One-Sensor Data

In the following simulations, we conducted tests on features extracted from both Fourier-based and wavelet-packet-based for assessing the applicability of wavelet-packet-based analysis as a tool for vibration monitoring. The Fourier-based features are defined as the power spectrum of a 256-point signal segment, and the result is a 129–dimensional vector where each component corresponds to one of the 129 uniform frequency band energies.

The two feature selection processes, *Approaches I* and *II* as described in Section III-C, are applied on both wavelet-packet-based feature components and Fourier-based feature components to select the best discriminant feature components. The obtained feature components are then used as input to train the neural network classifier. For each of the feature selection approaches, the eight highest discriminant ($d = 8$) feature components (out of 254) are used to form the final feature vector. d is chosen based upon LDA where only the eight highest eigenvalues are dominant in the given data set. Table II provides the dimension of the final feature vector for the two approaches. In the following, the feature selection method *Approach I*, as described in Section III-C, is designated as PWM, while the

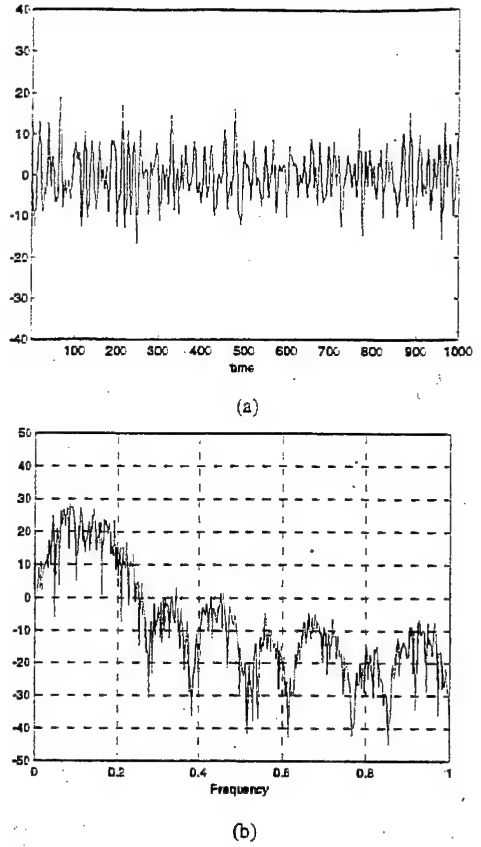


Fig. 11. Color noise and its power spectrum. (a) Color noise. (b) PSD of color noise.

Approach II is designated as KNK. In general, the computation cost for PWM will be less than that of KNK.

The network architecture is D-D-8, where D is the dimension of the final feature vector. In the training process, the network is trained until the mean-square error is below 0.01, or the maximum number of epochs (10 000) is reached. In practice, the neural network will not produce a perfect decision, i.e., only one 1 in the output neuron while others are all 0's, and might produce values between zero and one. Hence, it was decided to use the maximum output value as the most likely fault condition (i.e., a hard decision). In all simulations, a clear winner can always be identified.

The classification results are shown in Tables III–VI. Note that the unit of error is % in all classification results. "*Tr. Err.*" is referring to the training error, while "*Test Err.*" is referring to the testing error. Note that the performance of using different sensor data shows significant differences. For example, the testing errors of using data from sensors 5, 6, and 8 are relatively higher than those of other sensors for both the wavelet-packet-based and Fourier-based approaches. It is, thus, inferred that some sensors are not sensitive to the detection of specific fault symptoms. This suggests the need to use multiple-sensor data to search for class-specific features.

F. Test Result Using Eight-Sensor Data

In the following tests, feature components from all eight sensors are all used to begin the feature selection process; i.e., the comparison of discriminant powers is conducted on features coming from all eight-sensor data.

Table VII provides the dimension of the final feature vector for the two approaches based on wavelet packet features and Fourier features, respectively. The number of features chosen are with respect to ranking order in the discriminant function, as discussed in Section III-C. All simulation settings, network architecture and MSE goal, are the same as previous tests. The classification results are displayed in Tables VIII and IX corresponding to feature selection methods PWM and KNK, respectively. In the tables which follow, FT refers to the Fourier-based features while WPT refers to wavelet-packet-based features. The results show the performance is much improved when combining data from all sensors. If we use only one sensor, the crucial information for the specific fault symptom may not be detected and the overall classification performance may be lower. This confirms a general understanding that some fault symptoms can only be detected by some neighboring sensors. To qualitatively analyze the parameter d with respect to classification performance remains to be investigated in the near future.

Additionally, it is observed that the performance of the Fourier-based approach shows slightly better results than the wavelet-packet-based approach. It is concluded that features providing discriminant information may demonstrate narrow-band frequency characteristics in this data set. In such cases, the Fourier-based approach is ideally the better candidate for extracting signal features. Recall that there is a slight amount of frequency overlap among the wavelet basis functions, thus, a particular frequency may be sensed by two different basis functions. This frequency leakage may lead to worse performance using wavelet-packet-based features. Nevertheless, the WPT is still able to extract the essential discriminant features and achieve a satisfactory performance.

G. Test on Data Corrupted by Additive White Noise

A measured vibration signal can be considered to have the following components: the fault response caused by faulty equipment; vibration from normal machine components; vibration of neighboring machinery; and measurement variation. In monitoring vibration signals, we considered the *noise* to consist of vibration from machine components (other than the faulty response), neighboring machinery, and measurement noise. The presence of noise complicates the monitoring tasks in two forms: by masking the signal of interest and by increasing the vibration values beyond monitoring criteria, when, in fact, the component being monitored experiences no sign of malfunction. Note the original Westland data were collected fom a laboratory test stand. The data sets are very clean. To test further the feasibility of the wavelet-packet-based feature extraction technique on the presence of noise (a realistic environment), simulated data were artificially generated by adding different types of noise to the original vibration signals [9]. The goal was to investigate the robustness of wavelet-packet-based features when the data was subjected to the presence of noise. In the following simulations, the signals are first corrupted with artificially generated noise under different SNR, then the WPT and FT are applied on the corrupted signal to obtain the signal's

TABLE XIV
CLASSIFICATION PERFORMANCE (COLOR NOISE; SNR = 0 dB; PWM)

		d=1	d=2	d=3	d=4	d=5	d=6	d=7	d=8
WPT	Tr. Err.	12.5	0	0.25	0	0	0	0	0
	Test Err.	13.08	0.5	0.25	0.08	0.5	0.25	0.67	1.08
FT	Tr. Err.	15.75	2.5	0.75	0.75	0.25	0.25	0	0.25
	Test Err.	18	5.92	9.5	6.83	4.75	5.5	4.33	3.75

TABLE XV
CLASSIFICATION PERFORMANCE (COLOR NOISE; SNR = 0 dB; KNK)

		d=1	d=2	d=3	d=4	d=5	d=6	d=7	d=8
WPT	Tr. Err.	0.5	0.75	0	0	0	0	0	0
	Test Err.	3	2.42	1.17	0.56	0.42	2.25	0.75	1
FT	Tr. Err.	2.25	1.5	0.75	0.75	0	0.75	0	0
	Test Err.	7.08	7.08	2.42	3.75	3.25	2.17	2.83	4.17

TABLE XVI
CLASSIFICATION PERFORMANCE (COLOR NOISE; SNR = −3 dB; PWM)

		d=1	d=2	d=3	d=4	d=5	d=6	d=7	d=8
WPT	Tr. Err.	12.5	0	0	0	0	0	0	0
	Test Err.	13.08	0.08	0.08	0.42	0.25	0.25	0.17	0.25
FT	Tr. Err.	27	4.5	2	1.25	1.25	0.5	0.25	0.5
	Test Err.	30.25	10.83	16.08	14.42	13.08	12.08	11.92	13.17

TABLE XVII
CLASSIFICATION PERFORMANCE (COLOR NOISE; SNR = −3 dB; KNK)

		d=1	d=2	d=3	d=4	d=5	d=6	d=7	d=8
WPT	Tr. Err.	0.25	0.25	0	0	0	0	0.25	0
	Test Err.	0.75	0.83	0.58	0.67	1.58	0.33	1.67	2.33
FT	Tr. Err.	7.25	3.25	2.25	1	0.5	1.75	0.25	1
	Test Err.	10	9.83	9.33	11.17	10	13.33	10.92	11.83

time–frequency feature. Finally, the proposed feature selection method is used to identify discriminant feature components that will be used as input to train a neural network classifier. In this study, we use three types of noise to model the vibration signal other than the signal being monitored.

The first type of noise model used is white Gaussian noise, where no frequency is dominating, as shown in Fig. 10. This noise has an AR(0) model

$$X_k = a_k \tag{4.1}$$

where a_k are normally distributed with zero mean and variance σ_a^2. In this study, we use the Matlab function randn( ) to generate a_k.

Tables X–XIII show the results under different SNR. The results reveal that the wavelet-packet-based approach demonstrates better results than the Fourier-based approach. It was also observed that the difference of performance between the wavelet packet approach and the Fourier-based approach is even higher when the noise power is increased.

H. Test on Data Corrupted by Additive Color Noise

The second type of noise used to corrupt original data is colored noise where a group of frequencies is dominant, as shown in Fig. 11. Such a noise can be generally represented by an ARMA$(n, n-1)$ model [26]

$$\begin{aligned} X_k = {} & \phi_1 X_{k-1} + \phi_2 X_{k-2} + \cdots + \phi_n X_{k-n} \\ & + a_k - \theta_1 a_{k-1} - \theta_2 a_{k-2} - \cdots - \theta_{n-1} a_{k-n+1}. \end{aligned} \tag{4.2}$$

Coefficients ϕ_i and θ_i determine the center frequency and bandwidth of the noise. The a_k are normally distributed with zero mean and variance σ_a^2. In our tests, however, we generated such a noise by convolving a white-noise sequence with a bandpass filter. We generated the colored noise such that the dominant frequencies lie between the digital frequency band 0 to 0.25π. This is the band where the original signal contains most of its energy, as can be seen from Fig. 9. Tables XIV through XVII show the classification results of conducted simulations corresponding to different SNR. In all simulations, better results are obtained via the wavelet-packet-based approach.

TABLE XVIII
CLASSIFICATION PERFORMANCE (PINK NOISE; SNR = 0 dB; PWM)

		d=1	d=2	d=3	d=4	d=5	d=6	d=7	d=8
WPT	Tr. Err.	0.5	0	0	0	0	0	0	0
	Test Err.	0.5	0.08	0.67	0	1.33	0.08	0.33	0.08
FT	Tr. Err.	0.5	1.25	0	0.25	0	0	0	0
	Test Err.	1.33	2.58	2.25	1.5	2.5	0.92	0.42	0.5

TABLE XIX
CLASSIFICATION PERFORMANCE (PINK NOISE; SNR = 0 dB; KNK)

		d=1	d=2	d=3	d=4	d=5	d=6	d=7	d=8
WPT	Tr. Err.	0	0.25	0.25	0	0.25	0	0	0
	Test Err.	0.83	0.92	0.17	0.42	0.25	0.58	0	0.17
FT	Tr. Err.	1.75	0.5	0.5	0.25	0	0.25	0.25	0
	Test Err.	3.58	4.58	1.92	2	1.91	0.67	1.42	0.83

TABLE XX
CLASSIFICATION PERFORMANCE (PINK NOISE; SNR = −3 dB; PWM)

		d=1	d=2	d=3	d=4	d=5	d=6	d=7	d=8
WPT	Tr. Err.	0.5	0.25	0	0	0	0.25	0.25	0.25
	Test Err.	1.83	0.33	0.33	0.08	0.5	0.42	1.42	0.42
FT	Tr. Err.	2.5	1	0.75	0	0.5	0	0	0
	Test Err.	5.67	3.25	4.08	3.17	4.17	1.33	2.5	1.17

TABLE XXI
CLASSIFICATION PERFORMANCE (PINK NOISE; SNR = −3 dB; KNK)

		d=1	d=2	d=3	d=4	d=5	d=6	d=7	d=8
WPT	Tr. Err.	0.75	0	0	0	0	0.25	0.25	0.25
	Test Err.	1.25	0.17	1.25	0.5	0.17	0.75	0	0.17
FT	Tr. Err.	1.25	1.25	0.5	0.75	0.5	0	0	0
	Test Err.	7	5.25	3.5	4.17	3.08	3.42	3.17	2.83

I. Test on Data Corrupted by Additive Pink Noise

The third type of noise employed is pink noise, where power decreases as frequency increases, as depicted in Fig. 12. It can be expressed by an AR(1) model

$$X_k = \phi_1 X_{k-1} + a_k \tag{4.3}$$

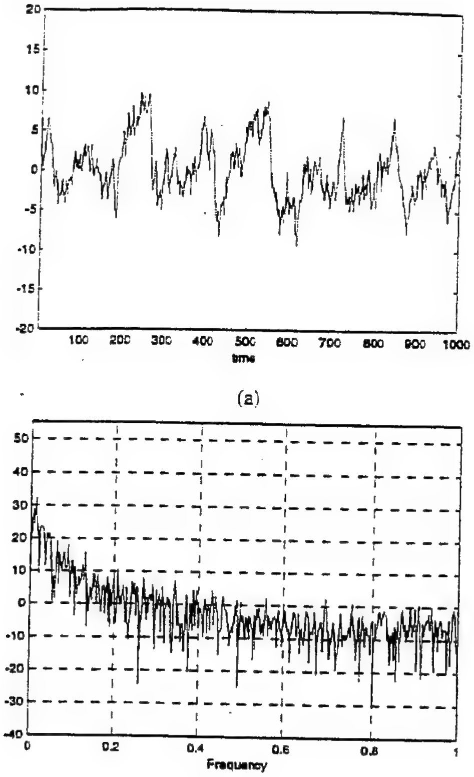
where a_k are normally distributed with zero mean and variance σ_a^2. In the test, ϕ_1 is set to be 0.95, and resulting noise is displayed in Fig. 12. The test results are shown in Tables XVIII–XXI. Again, it is confirmed that the wavelet-packet-based approach produces better results.

J. Discussion of Test Results

By examining Tables III and IV, where only one sensor is used for the searching of class-specific feature components, it is clear that some sensors provide little class separability information in the sense of frequency analysis. This indeed confirmed our understanding that the faulted symptom is often localized and can only be detected by neighboring sensors. It suggests that data collected from multiple sensors will provide better classification information and lead to better performance. From the results of simulation on the *original data set*, it is observed that no improvement is made through the wavelet-packet-based approach on this data set and, in several cases, it is even slightly worse than the Fourier-based approach. As mentioned before, this could be due to the "overlap" of frequency content among wavelet packet basis functions. Nevertheless, the wavelet-packet-based approach shows very promising results in a realistic environment for which the data are corrupted by noise.

V. CONCLUSION

This paper has investigated the feasibility of applying the WPT to the classification of vibration signals. Using the WPT, a rich collection of time-frequency characteristics in a signal can be obtained and examined for classification purposes. In this paper, we detailed an innovative feature selection process that exploits signal class differences in the wavelet packet node energy. This results in a reduced-dimensional feature space compared to the dimension of the original time-series signal. The wavelet-packet-based features, obtained by our method for vibration signals, yields nearly 100% correct classification when used as input to a neural network classifier.



(a)

(b)

Fig. 12. Pink noise and its power spectrum. (a) Pink noise. (b) PSD of pink noise.

In Section II, we reviewed the Fourier-based analysis on the extraction of frequency information from a signal and discussed the possible inherent drawbacks due to its fixed time-frequency resolution. The WPT that overcomes the fixed time-frequency resolution was then presented. To alleviate the *time-variant* characteristics of the WPT coefficients, wavelet packet node energy was used as an essential time-frequency feature measure of the signal. Although the wavelet packet node energy provided us with a multiresolution view of a signal, it simultaneously introduced a higher dimension space compared to the original time-domain signal. To reduce the dimensionality, it was shown that LDA had some practical problems when the feature dimension was relatively high compared to the number of collected samples since it involved calculation of the inverse of the covariance matrix. In such a case, two-feature selection criteria based on measures of the overlap of the conditional probability density function among different classes were proposed to avoid the possible numerical problems as presented in Section III. In Section IV, the proposed wavelet-packet-based classification system, which combined a wavelet-packet-based feature extractor and a neural network classifier, was tested on a real data set known as the Westland data set. Numerically, it was observed that significant improvement can be achieved when using multiple sensor data. This validated our understanding that a faulted symptom is localized and can only be detected by the neighboring sensors. Both the Fourier-based features and wavelet-packet-based features achieved excellent classification results on the original Westland data set when all eight sensor were utilized. Nonetheless, the improved time-frequency resolution of the WPT is significant when we are confronted with signals corrupted by artificially synthesized noises. In the extended tests, the wavelet-packet-based approach showed very promising results compared to the Fourier-based approach.

References

[1] G. G. Yen, "Health monitoring of vibration signatures in rotorcraft wings," *Neural Processing Lett.*, vol. 4, no. 3, pp. 127–137, Dec. 1996.

[2] R. J. Ferlez and D. C. Lang, "Gear-tooth fault detection and tracking using the wavelet transform," in *Proc. 52nd Meeting Society for Machinery Failure Prevention Technology*, Mar. 1998, pp. 451–460.

[3] J.-C. Lo *et al.*, "Fault prediction in transmissions using wavelet analysis," in *Proc. 52nd Meeting Society for Machinery Failure Prevention Technology*, Mar. 1998, pp. 441–450.

[4] P. D. Samuel *et al.*, "Fault detection in the OH-58A main transmission using the wavelet transform," in *Proc. 52nd Meeting of the Society for Machinery Failure Prevention Technology*, Mar. 1998, pp. 323–335.

[5] B. Liu *et al.*, "Machinery diagnosis based on wavelet packets," *J. Vibration Contr.*, vol. 3, no. 1, pp. 5–17, Jan. 1997.

[6] J. E. Lopez and K. Oliver, "Overview of wavelet/neural network fault diagnostic methods applied to rotating machinery," in *Proc. Joint Conf. Technology Showcase Integrated Monitoring, Diagnostics and Failure Prevention*, Apr. 1996, pp. 405–417.

[7] B. E. Parker *et al.*, "Helicopter transmission diagnostics using vibration signature analysis," in *Proc. Joint Conf. Technology Showcase Integrated Monitoring, Diagnostics and Failure Prevention*, Apr. 1996, pp. 419–430.

[8] M. A. Essawy, S. Diwakar, and S. Zein-Sabatto, "Fault diagnosis of helicopter gearboxes using neuro-fuzzy techniques," in *Proc. 52nd Meeting Society for Machinery Failure Prevention Technology*, Mar. 1998, pp. 293–303.

[9] D. Paul, "Condition monitoring and defect diagnosis in manufacturing process using DDS and wavelets," Ph.D. dissertation, Dep. Elect. Eng., Michigan Technol. Univ., Houghton, MI, 1995.

[10] O. Rioul and M. Vetterli, "Wavelets and signal processing," *IEEE Signal Processing Mag.*, vol. 8, pp. 14–38, Oct. 1991.

[11] F. Hlawatsch and G. F. Boudreaux-Bartels, "Linear and quadratic time-frequency signal representations," *IEEE Signal Processing Mag.*, vol. 9, pp. 21–67, Apr. 1992.

[12] R. R. Coifman and M. V. Wickerhauser, "Entropy-based algorithms for best basis selection," *IEEE Trans. Inform. Theory*, vol. 38, pp. 713–718, Mar. 1992.

[13] A. Papoulis, *The Fourier Integral and Its Applications.* New York: McGraw-Hill, 1962.

[14] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis," *IEEE Trans. Inform. Theory*, vol. 36, pp. 961–1005, Sept. 1990.

[15] ——, "Orthonormal bases of compactly supported wavelets," *Commun. Pure Appl. Math.*, vol. XLI, no. 4, pp. 909–996, 1988.

[16] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 674–693, July 1989.

[17] A. N. Akansu and R. A. Haddad, *Multiresolution Signal Decomposition: Transforms, Subbands, Wavelets.* New York: Academic, 1992.

[18] M. V. Wickerhauser, *Adapted Wavelet Analysis from Theory to Software.* Natick, MA: Wellesley, 1994.

[19] P. A. Devijver and J. Kittler, *Pattern Recognition—A Statistical Approach.* London, U.K.: Prentice-Hall, 1982.

[20] K. Fukunaga, *Introduction to Statistical Pattern Recognition.* New York: Academic, 1992.

[21] J. Kittler, "Mathematical methods of feature selection in pattern recognition," *Int. J. Man-Mach. Studies*, vol. 7, no. 5, pp. 609–637, Sept. 1975.

[22] S. Watanabe and T. Kaminuma, "Recent developments of the minimum entropy algorithms," in *Proc. Int. Conf. Pattern Recognition*, 1998, pp. 536–540.

[23] R. P. Lippmann, "Pattern classification using neural networks," *IEEE Commun. Mag.*, vol. 27, pp. 47–64, Nov. 1989.

[24] B. G. Cameron, "Final report on CH-46 aft transmission seeded fault testing," Westland Helicopters, Ltd., U.K., Res. Paper RP907, Sept. 1, 1993.

[25] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1989.

[26] S. M. Pandit and S. M. Wu, *Time Series and Systems Analysis with Applications*. New York: Wiley, 1983.

Gary G. Yen (S'87–M'88–SM'97) received the Ph.D. degree in electrical and computer engineering from the University of Notre Dame, Notre Dame, IN, in 1992.

He is currently an Associate Professor in the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater. Before he joined Oklahoma State University in 1997, he was with the Structure Controls Division, U.S. Air Force Research Laboratory, Albuquerque, NM. His research is supported by the U.S. Department of Defense, U.S. Department of Energy, U.S. Environmental Protection Agency, National Aeronautics and Space Administration, National Science Foundation, and the petrochemical and process industry. His research interests include intelligent control, computational intelligence systems, conditional health monitoring, wavelet theory, signal processing, and their defense applications. He served as the Local Arrangements Chair for the 1999 American Control Conference held in San Diego, CA.

Dr. Yen was an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS and *IEEE Control Systems Magazine* during 1994–1999. He served as Publicity Chair for the 1997 IEEE Conference on Decision and Control held in San Diego, CA, as Finance Chair for the 1998 IEEE International Symposium on Intelligent Control held in Gaithersburg, MD, as Finance Chair for the 1999 IEEE Conference on Control Applications and 1999 IEEE International Symposium on Computer Aided Control System Design, and as Program Chair for the 2000 IEEE Conference on Control Applications to be held in Anchorage, AK. On behalf of the IEEE Control Systems Society, he is a Representative to the IEEE Neural Networks Council Administrative Committee.

Kuo-Chung Lin received the B.S. degree in electrical engineering from Tamkang University, Tansui, Taiwan, R.O.C., and the M.S. degree in electrical and computer engineering from Oklahoma State University, Stillwater, in 1990 and 1998, respectively.

From 1997 to 1998, he was a Research Assistant in the Intelligent System and Control Laboratory, Oklahoma State University. He is currently with the IBM Server Development Group, Austin, TX. His research interests include wavelets, image/signal processing, pattern recognition, and neural networks.

APPENDIX B:

Winner Take All Experts Network For Sensor Validation

by

Gary G. Yen and Wei Feng

ISA Transactions, 40(2), 2001, pp. 99-110

ELSEVIER

ISA Transactions 40 (2001) 99–110

ISA TRANSACTIONS®

www.elsevier.com/locate/isatrans

Winner take all experts network for sensor validation

G.G. Yen *, W. Feng

Intelligent Systems and Control Laboratory, School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078-5032, USA

Abstract

The validation of sensor measurements has become an integral part of the operation and control of modern industrial equipment. The sensor under harsh environment must be shown to consistently provide the correct measurements. Analysis of the validation hardware or software should trigger an alarm when the sensor signals deviate appreciably from the correct values. Neural network based models can be used to on-line estimate critical sensor values when neighboring sensor measurements are used as inputs. The underlying assumption is that the neighboring sensors share an analytical relationship. The discrepancy between the measured and predicted sensor values may then be used as an indicator for sensor health. The proposed Winner Take All Experts (WTAE) network based on a 'divide and conquer' strategy significantly reduces the computational time required to train the neural network. It employs a growing fuzzy clustering algorithm to divide a complicated problem into a series of simpler sub-problems and assigns an expert to each of them locally. After the sensor approximation, the outputs from the estimator and the real sensor readings are compared both in the time domain and the frequency domain. Three fault indicators are used to provide analytical redundancy to detect the sensor failure. In the decision stage, the intersection of three fuzzy sets accomplishes a decision level fusion, which indicates the confidence level of the sensor health. Two data sets, the Spectra Quest Machinery Fault Simulator data set and the Westland vibration data set, were used in simulations to demonstrate the performance of the proposed WTAE network. The simulation results show the proposed WTAE is competitive with or even superior to the existing approaches. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Sensor validation; Neural network; Winner take all; Divide and conquer

1. Introduction

Sensor validation has become an essential part of the operation and control of modern industrial equipment, which completely relies on the correct sensor measurements. On one hand, in order for a situation to be fully comprehended and controlled, reliable data acquisition and interpretation is of the utmost importance in modern decision-making process. On the other hand, an increasing functionality in control system makes it further complicated and costly to accomplish the task of sensor validation and diagnosis [1]. Control systems for critical plants, whose operations must not be interrupted for safety reasons, are often configured with redundant sensors to provide fault tolerance and to ensure the required degree of safety. As a result, the redundant sensors once again increase

* Corresponding author. Tel.: +1-405-744-7743; fax: +1-405-744-9198.
E-mail address: gyen@ceat.okstate.edu (G.G. Yen).

0019-0578/01/$ - see front matter © 2001 Elsevier Science Ltd. All rights reserved.
PII: S0019-0578(00)00047-1

the cost and weight of the whole system. To achieve substantial savings on hardware redundancy, and, at the same time, to meet the requirements of reliable and accurate sensor measurements for the modern control systems, an intelligent sensor validation scheme based upon *analytical redundancy* is proposed in this paper.

Neural network based estimators can be used to observe critical sensor values when multiple neighboring sensor measurements are used as additional inputs. This framework is based upon the assumption that physically close sensors are analytically related. The estimated sensor value may also be used as synthetic data in the event of a sensor failure, and is often called a soft sensor or inferential sensor. Application of fuzzy set theory for the decision making process is advantageous, because the soft decision boundaries in fuzzy logic environments result in flexible, more human-like decisions. The information that the validation system deals with is often fuzzy, obscure rather than precise in nature, and fuzzy set theory allows us to model this imprecision appropriately and later permits us to reason in a linguistic language [2]. The proposed sensor validation scheme, synergistically integrates the neural network and fuzzy logic, provides a reliable health indicator for the critical sensor of interest.

For the completeness of the presentation, the remainder of this paper is organized as follows. In Section 2, we introduce the proposed Winner Take All Experts (WTAE) network architecture and the validation algorithm along with a brief literature review. In Section 3, we analyze the performance of the architecture on two benchmark problems. Finally, Section 4 provides the conclusion of the article and some pertinent observations.

2. The Winner Take All Experts network

A sensor is declared faulty when it displays a non-permitted deviation from the characteristic properties of the objects it is monitoring. In most cases, sensor validation scheme uses a number of hardware sensors provided a redundancy of information monitoring each important system parameter, from which a more reliable value was extracted. Although hardware redundancy is popular and solves many sensor validation problems, it possesses some obvious disadvantages [3].

- The expense of the redundant sensors, and the installing, maintaining of them for each important observation greatly increase the cost of the system.
- Even redundant sensors are in-place: all are sensitive to the common failures and likely to fail.

The technology from artificial intelligence helped to establish Knowledge Based System (KBS) that diagnoses sensors utilizing system operational knowledge in validating sensor values [4]. The knowledge-based system is efficient and helpful to yield valuable clues to the fault symptoms and their locations. But it demands an accurate model with a comprehensive understanding of the basic physics and the knowledge of all the potential variables, which are often unavailable, if not impossible, in most cases.

The primary tracking algorithms used to estimate the nominal sensor measurement are Kalman Filter and Extended Kalman Filter. The fundamental assumptions of the Kalman Filter are that the linear plant equations are known and have zero mean white noise with known covariance. These constraints make this approach difficult to model nonlinear system and vulnerable to correlated noise. The Extended Kalman Filter tries to linearize the nonlinear functions, but it is sensitive to the accuracy of the initial conditions [5].

Neural network based models can be used to estimate critical sensor values when other sensors' measurements are used as inputs. The basic premise behind this framework is that the sensed plant variables are not independent of each other [6]. The most popular neural network in use today is the Multi-Layer Perceptron (MLP) network. The feasibility of using MLP network in sensor validation has been studied since 1990 [7,8]. A modified MLP network was implemented for hardware based on-line learning soft sensor in 1998 [9].

A Radial Basis Function (RBF) neural network was investigated by Wheeler and Dhawn [10] for sensor signal tracking. The network used the k-means clustering algorithm for placement of the basis function centers. The result showed that RBF network is good at tracking the sensor signal when it varies slowly, but a general RBF network could not accurately learn the data and became unstable when the dynamics becomes more complex.

Traditional neural networks such as MLP and RBF networks have proved successful as universal function approximators and have been used in various problems, but the training algorithms are typically too slow for solving real-world problems in real time. In addition, when the problem becomes complicated, most of the systems could not even converge to a local minimum in a reasonable time due to hardware limitation and the inefficiency of the learning rule. Motivated by such concerns, a number of researchers have investigated methods of function approximation incorporating ideas from the communities of statistics and artificial intelligence [11]. The general approach is to divide a complicated problem into several simpler sub-problems and assign a function approximator or 'expert' to each sub-problem locally [12].

If a set of training cases may be naturally divided into subsets that correspond to distinct subtasks, interference can be reduced by using a system composed of several different 'expert' networks plus a fuzzy membership clustering network that decides which of the experts should be used for each training case [13]. A system of this kind can be used only when the division into subtasks is known prior to training (e.g. along the operating regimes or trajectories). The proposed WTAE network is a modular architecture that works on the principle of 'divide and conquer'. The model employs a growing fuzzy clustering method to divide the input space into overlapping operating regions on which 'experts' act, and a fuzzy membership clustering network to weight these experts to form an overall network output. The idea behind such a system is that the growing fuzzy clustering algorithm allocates a new case to one of the experts, and, if the output is incorrect, the weight adaptations are localized to this expert.

2.1. Network architecture

The proposed WTAE network architecture is shown in Fig. 1. Without loss of generality, each expert network is a typical single hidden layer MLP network. The networks input vector consists of the signals from each related sensor that is located nearby the critical sensor of interest. The outputs from every expert network form an output vector $y = [y_1, y_2, ..., y_M]$, where M denotes the number of expert networks. The vector is simultaneously fed into the fuzzy membership clustering network, which produces the membership level, $u_i(x) \in [0, 1]$, where $i = 1, ..., M$. Each membership level corresponding to each expert network altogether forms a membership vector $\mu = [\mu_1, \mu_2, ..., \mu_M]$.

The fuzzy membership clustering network is a single layer network with a Gaussian output non-linearity. Let the ith center and the ith variance of the fuzzy membership level network be $C_i = [C_{i1}, C_{i2}, ..., C_{iN}]$ and $\sigma_i = [\sigma_{i1}, \sigma_{i2}, ..., \sigma_{iN}]$, respectively, where N is the dimension of input space. The corresponding output $\mu_i(x)$ after the Gaussian non-linearity is given by:

$$\mu_i(x) = \prod_{j=1}^{N} \mathrm{e}^{\frac{(x_j - c_{ij})^2}{2\sigma_{ij}^2}}. \tag{1}$$

The Gaussian function is to ensure that $\mu_i \in [0, 1]$, and if x belongs to ith cluster,

$$\|x - C_i\| < \|x - C_j\| \Longleftrightarrow \mu_i > u_j, \tag{2}$$

for $j = 1, 2, ..., i-1, i+1, ..., M$. $\|\|$ is referred to as the Euclidean norm. Thus, the ith expert's influence is localized to a region around C_i. A Winner Take All function is denoted by $a = \mathrm{compet}(\mu)$, which takes one input argument μ, and returns an output row vector with 1 at the ith element having maximum membership value $i = \underset{j}{\mathrm{argmax}}(\mu_j(x))$, and 0 elsewhere. The Winner-Take All function selects the winner expert network outputs to form the overall estimated outputs by $\hat{y}(x) = y \cdot a^{\mathrm{T}}$.

Finally, the output from the estimator and the real sensor are compared both in the time domain

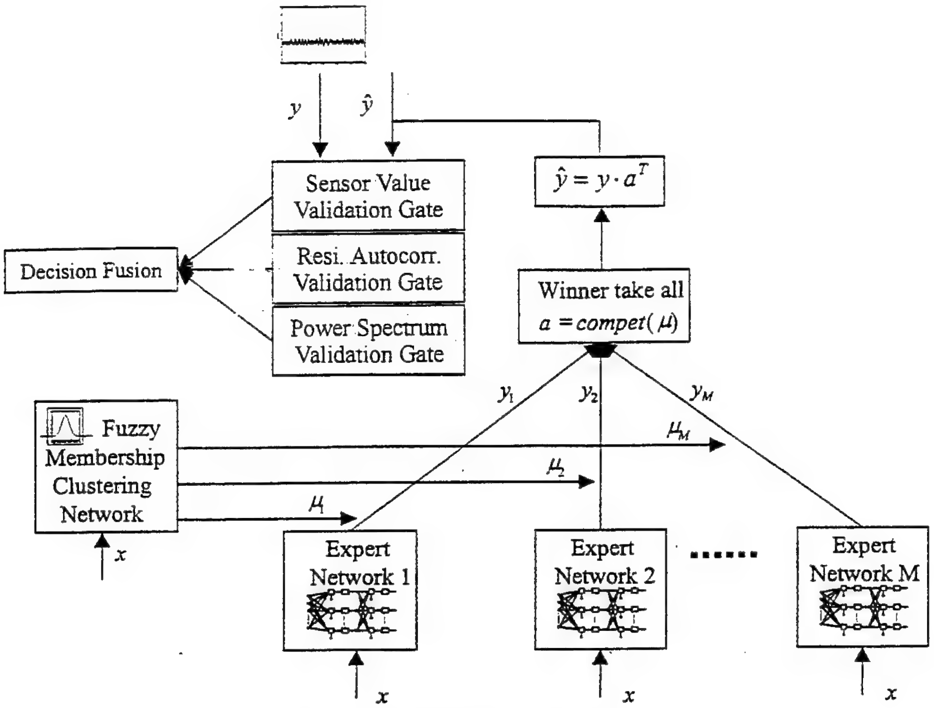

y
ŷ
Sensor Value Validation Gate
Resi. Autocorr. Validation Gate
Power Spectrum Validation Gate
Decision Fusion
ŷ = y · a^T
Winner take all
a = compet(μ)
y1
y2
yM
μM
μ2
μ1
Fuzzy Membership Clustering Network
x
Expert Network 1
Expert Network 2
Expert Network M
x
x
x

Fig. 1. WTAE network architecture.

and frequency domain. Three fault indicators are used to provide the necessary redundancy to identify sensor failure. The first validation gate, the sensor value validation gate, compares the tracking output directly with the real sensor data. In the second validation gate, the residual of the two time series is investigated using the autocorrelation coefficient. The power spectrum density of the two signals is compared in the last validation gate. A decision level fusion of the three validation gates is accomplished by the intersection of the three fuzzy sets (to be further discussed in Section 2.3). The output from the decision stage shows the sensor health confidence level of the critical sensor, which will serve as an indicator for the human operator to take necessary actions in order to remedy or ameliorate the sensor fault, if necessary.

2.2. Training algorithm

The WTAE network is trained using the Growing Fuzzy Clustering algorithm, where the data is trained sequentially. The network will continuously add local experts that contribute to the final estimate in the off-line training step. The growth criterion is essential that if the new expert contributes little to the output, then, not only does the complexity of the network increase unnecessarily, but it adds to the computational burden. This leads to the question of how the network growth must be regulated. Fig. 2 shows the training process used in the WTAE network, where the training data is received sequentially. The networks begin with no expert network. The first observation (x^1, t^1), where x^1 is the input and t^1 is the corresponding target output, is used to initialize the first fuzzy membership cluster characterized by (C_1, σ_1), by setting $C_1 = x^1$ and σ_1 a unity vector. The degree of belonging for input x^1 to the first cluster is measured by

$$\mu_1(x^1) = \prod_{j=1}^{N} e^{\frac{\left(x_j^1 - c_{1j}\right)^2}{2\sigma_{1j}^2}}. \tag{3}$$

At the same time, the first sample is stored locally in the first training set corresponding to the

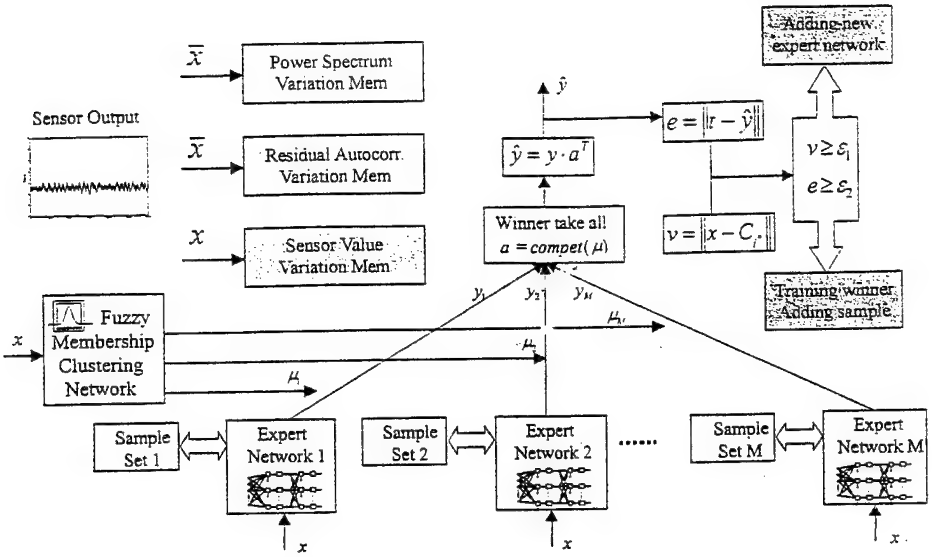


Fig. 2. WTAE network training algorithm.

first fuzzy cluster defined above.Next, a simple architecture MLP expert network is built up and trained by the first training set. The structure of the MLP is determined by the complexity of the problem. The training goal, which decides the stopping criteria for the MLP training, is determined by the mean square error between the estimator and the real sensor output (target) we can tolerate.

As observations are continuously received, the network grows by adding new expert networks, if necessary. The decision to add a new expert for an observation (x^k, t^k) depends on its novelty, for which the following two criteria must be met:

$$\left\|x^k - C_i^*\right\| \geqslant \varepsilon_1, \tag{4a}$$

$$e^k = \left\|t^k - \hat{y}^k\right\| \geqslant \varepsilon_2, \tag{4b}$$

where C_i^* is the nearest fuzzy cluster to x^k in the input space, $\hat{y}^k$ is the resulted output from WTAE network, and $\varepsilon_1, \varepsilon_2$ are user-specified thresholds. The first criterion decides that the input must be far away from the existing fuzzy clusters, and the second criterion says that the error between the network output and the target value must be significant. The criterion ε_1 represents the scale of resolution in the input space. On the other hand, the value ε_2 is chosen to represent the desired accuracy of the network output [14]. In practice, the trending approach should be considered to maintain the stability [15].

When a new, $M+1$th expert network is added to the WTAE, the parameters associated with this local expert network are assigned as follows:

$$C_{M+1} = x^k, \tag{5a}$$

$$\sigma_{M+1,1} = \sigma_{M+1,2} = \ldots, = \sigma_{M+1,N} = 1, \tag{5b}$$

$$\mu_{M+1} = \prod_{j=1}^{N} e^{\frac{\left(x_j^k - C_{M+1,j}\right)^2}{2\sigma_{M+1,j}^2}}, \tag{5c}$$

while the new sample set S_{M+1}:

$$S_{M+1} = \left\{x^k\right\}. \tag{6}$$

Also, the new expert network is built up and trained by the sample set S_{M+1}.

When the observation (x^k, t^k) does not satisfy the two novelty criteria, the Resilient Back-propagation training algorithm [16] is used to adapt the winning MLP expert network parameters, while other losing experts remain unchanged. The fuzzy center is updated by calculating the mean of the winning sample set S_i^*:

$$C_i^* = \frac{1}{R}\sum_{r=1}^{R} x_i^{r*}. \tag{7}$$

where R is the number of the samples in $S_i^* = \{x_i^1 .. s_i^2 .. ..., x_i^R .)$. The variance of the fuzzy cluster is determined by both the previous variance and the current statistic variance of the updated sample set $S_i^* = \{S_i^* . x^k\}$. We use a momentum to combine these two factors together:

$$\sigma_i^{*2}(t) = \gamma\sigma_i^{*2}(t-1) + (1-\gamma)\frac{1}{R}\sum_{r=1}^{R}(x^r - C_i^*)^2. \tag{8}$$

When the first several samples are added to the sample set S_i, the statistical variance will be close to zero. The momentum helps to avoid the risk of zero variance and also keep the statistical characteristics of the variance.

The overall output is calculated based on a winner take all rule. First, the whole output μ from the fuzzy clusters goes through the competition layer where each neuron excites itself and inhibits all the other neurons. The transfer function of the competition layer is defined as shown below

$$a = \text{compet}(\mu). \tag{9}$$

It works by finding the index i^* of the neuron with the largest net input, and setting its output to 1 (with ties going to the neuron with the lowest index). All other outputs are set to 0,

$$ai = \begin{cases} 1, i = i^* \\ 0, i \neq i^* \end{cases}, \quad \text{where } \mu_i^* \geqslant \mu_i, \ \forall i \neq i^*. \tag{10}$$

Next, the overall output from the estimator, $\hat{y}$, is calculated as follows.

$$\hat{y} = y \cdot a^{\mathrm{T}}. \tag{11}$$

After the winner is selected by the fuzzy membership function, the only MLP that has to be calculated is the winning expert network.

The last step is executed after all the training samples were trained. The number of samples in each sample set could be dramatically different. So. the purpose of the last step is pruning the expert that contains very little training samples and re-classifying these samples to nearest fuzzy cluster based upon Euclidean measures.

2.3. Sensor failure mode detection algorithm

The first part of the WTAE network provided us with the estimated critical sensor value when other sensors' measurements are used as additional inputs. To validate a sensor signal, the validation algorithm should trigger an alarm when the sensor signal significantly deviates from the corrected value.

In order to build up the validation gates, the characteristics of both the sensor and the monitored operating environment have to be comprehended. The validation scheme proposed in this paper contains both time domain and frequency domain sensor failure detection by using three fuzzy sensor validation gates. So, in the learning stage, the statistical information both in the time domain and the frequency domain has to be learned and recorded by three memories. The first one is used to store the minimum, maximum, and mean value of the sensor signal in the training data set. The other two memories are used to record the variation of autocorrelation coefficients r of the residual of the sensor signal, and power spectrum p of the time series. A time window was specified in order to calculate r and p, which are defined as follows:

Definition 1. For a series of data $\{x^k : k = 1, ..., K\}$, the nth auto-covariance coefficient is defined as:

$$g_n = \sum_{k=n+1}^{K} (x^k - \overline{x})(x^{k-n} - \overline{x})/K. \tag{12}$$

where $\overline{x}$ is the sample mean:

$$\overline{x} = \left(\sum_{k=1}^{K} x^k\right)/K. \tag{13}$$

Then the nth auto-correlation coefficient is

$$r_n = g_n/g_0. \tag{14}$$

Definition 2. For a series of data. $\{x^k : k = 1.\ldots.K\}$ can be represented by its Fourier series:

$$x^k = (1/K)\sum_{n=0}^{K-1} C_n e^{j2\pi nk/K}, \tag{15}$$

where C_n are the Fourier coefficients. Then, Power Density Spectrum Coefficient can be defined as:

$$p_n = \|C_n\|^2. \tag{16}$$

After each r and p corresponding to each time window in the learning data set is available, the time series, the auto-correlation coefficients, and the power spectrum density of the sensor signal are learned and analyzed with the eight commonly found sensor failure modes [16], as shown in Fig. 3. x-Axis and y-axis are referred to as the time and vibration accelerations in the first column (Sensor Signal), time and autocorrelation coefficients [defined in Eq. (14)] in the second column (Residual Autocorrelation), and frequency and power density spectrum coefficients [defined in Eq. (16)] in the third column (Power Spectrum). It is worth noting that by inspecting the signatures from different feature spaces, it is easier to identify a good sensor from a faulted one.

In Fig. 3, the first column shows the eight failure modes' signals compared to the nominal state signal in the time domain. The other two columns are used to compare the autocorrelation coefficients of the residual, and the power spectrum density for the interested frequency range (for example, in the Westland data set, 3–10 kHz). It is also clear from the figure that using signatures from only one feature space is not sufficient to detect the normal signal from all the other failure modes' signals. However, it is much robust to detect the faults by combining the comparison results from all three feature spaces. In order to measure the distances between objects or points in the feature space, a distance measure is used. There are a number of distance metrics that can be used as a tool to measure a similarity between vectors, for example, Euclidean distance, Mahalanobis distance, and Minkowski distance. Without loss of generality, to calculate the difference between the signatures, the Manhattan distance [20] is chosen as the distance between two vectors measured along orthogonal axes.

$$d = \|x_1 - y_1\| + \|x_2 - y_2\| + \ldots + \|x_n - y_n\|. \tag{17}$$

where $x = [x_1, x_2, \ldots, x_n]$, $y = [y_1, y_2, \ldots, y_n]$ are the two vectors to be measured.

To investigate the variation of the autocorrelation coefficients r for each time window, the Manhattan distances between each vector are calculated with result d_i, where $i = 1, \ldots, m$. $m = \frac{n(n-1)}{2}$. n is the number of the time window in the training data set. The minimum, maximum, and mean value of d_i are recorded as the variation characteristics of the autocorrelation coefficients r. The mean vector $\bar{r}$ is also recorded for comparison purpose in the detecting stage. Similar procedure is applied to the power spectrum density p.

Three Validation Gates, sensor value, residual autocorrelation, and power spectrum, were established to provide redundancy using the information from the learning stage. The basic structure of each gate (or membership function) is based on the dynamic fuzzy validation curve. The mean values from the three memories are set as the highest confidence value for the bell shape validation function. The minimum and the maximum values are set at 10% confidence value points as shown in Fig. 4.

In the detecting stage, the auto-correlation of the residual of the sensor output was calculated in every time window and was compared with the mean autocorrelation coefficients $\bar{r}$ in the Manhattan distance by the corresponding validation gate. The output will be the sensor health confidence level in autocorrelation feature domain. A similar approach is applied to the power spectrum density and each sensor output datum. Finally, the three outputs from the three gates are fused by fuzzy intersection

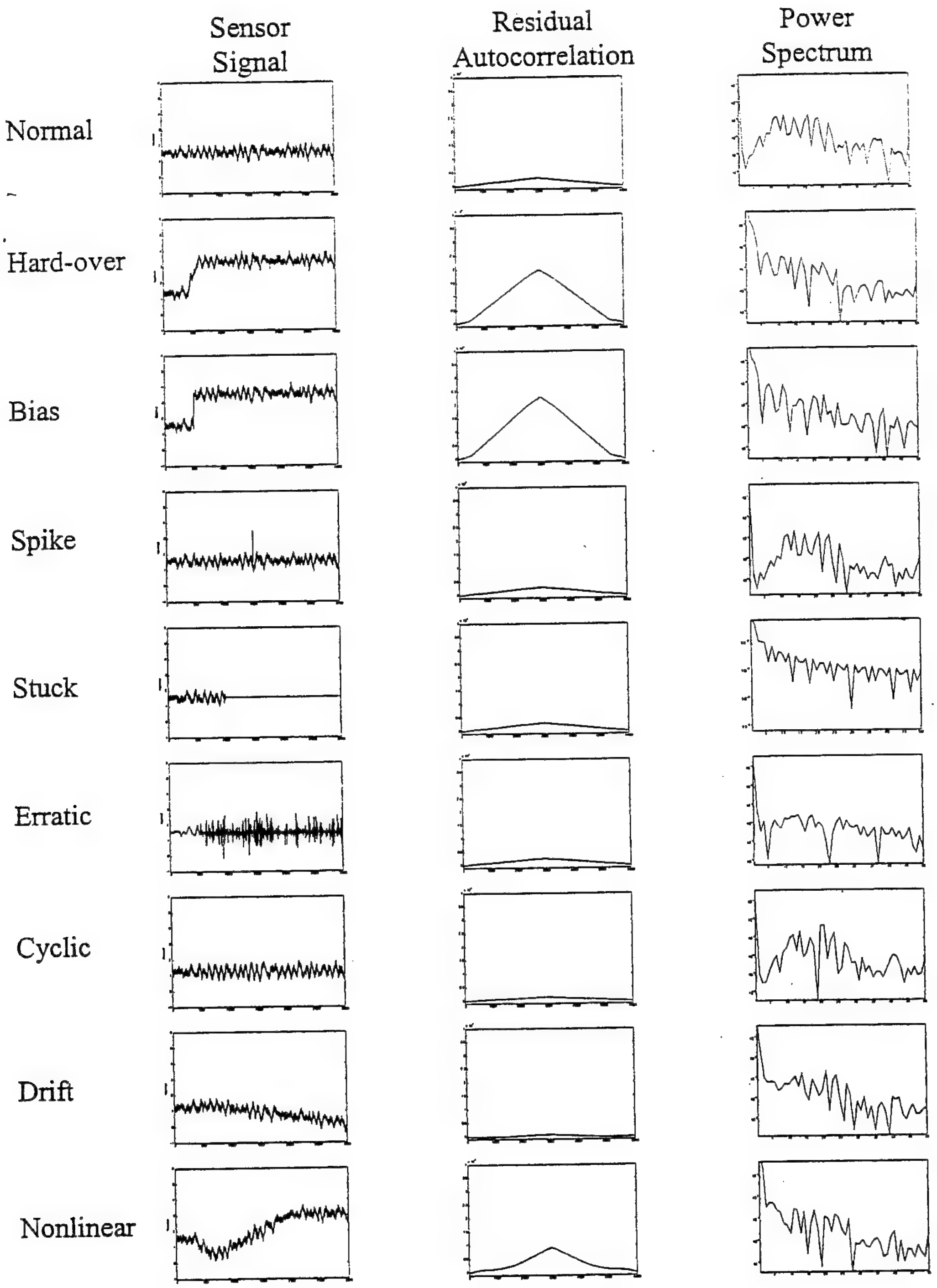


Fig. 3. Feature spaces of sensor signals.

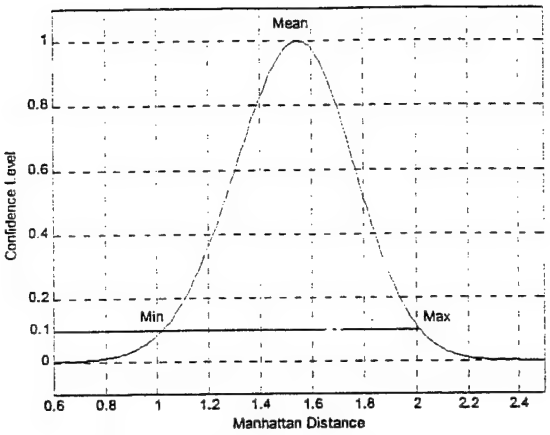


Fig. 4. A bell shape fuzzy validation gate.

with the fuzzy min operator. The decision level fusion output is a final confidence level of the sensor health.

3. Simulation results

Simulation was used to demonstrate the expected performance of the WTAE networks. Two data sets were used for training and testing the sensor validation system in our studies. The first benchmark data set was generated from a Spectra Quest (SQ) Machinery Fault Simulator. The second data set was a time-series vibration data set, commonly known as Westland vibration data.

3.1. SQ Machinery Fault Simulator data set

This data set consists of vibration data recorded from the SQ Machinery Fault Simulator. The instrument is constructed with special kinds of bearings, rotors with split collar ends, and a split bracket bearing housing. The simulator offers a wide range of predictive maintenance and the signatures of various bearing faults.

Here, we use three accelerometer sensors in proximity as the inputs x to the WTAE network, which have the similar high frequency waveform, to approximate the outputs y of the fourth sensor measurements of interest. System parameters, μ_i, σ_i, and C_i are referred to the statistical distribution of the ith fuzzy expert network. The relationship of the sensors is obviously nonlinear, and the modeling work is very complex. The sample data set separated into two parts. The first 3000 observation pairs are used as the training set, and the remaining 3000 points serve as the testing set. The WTAE network was trained and compared with MLP and RBF based estimators.

The performance function for the experts is the mean square error (MSE) between the network outputs and the target output. A 10 hidden nodes MLP structure was chosen ad hoc to be the local expert network with a Resilient Back-propagation training algorithm. After the WTAE network had been trained for 41 min, the MSE of the training set was 0.0234. The resulting network incorporated 23 local experts, with a pruning criterion of 10 samples per sample set. The mean square error of the 3000 testing samples using WTAE networks was 0.1631. The thresholds, ε_1 and ε_2, are chosen to be 0.1 and 0.001, respectively.

The same training set was also applied for training a MLP network. The total number of nodes in the MLP was estimated by the Vapnik–Chervonenkis (VC) dimension theory [17,18] with a conservative upper bound 300. After training several pre-selected structures, the network with the best result was a MLP with a 150-node hidden layer. This network was trained by Resilient Back-propagation algorithm as well. After 41 min, the MSE was 0.0554. Also, the networks were tested by the same 3000 samples as the WTAE networks did. The resultant MSE is 5.9529.

The third estimation method used for comparison is the RBF network. Here, we used the k-Means Clustering Algorithm to train the first layer and the Moore–Penrose algorithm for the second layer [10]. Because of the high computational cost involving in matrix inversion, the RBF networks can only be trained by 775 samples in 41 min. The resulting MSE for the training set is 0.0550. The number of first layer clusters is 339 because of the complexity of the input space. The next 775 samples were tested with a MSE of 0.0787.

From the above simulation result, we can draw the following conclusions. Just like MLP and RBF based neural network, WTAE can uniformly approximate any continuous function without knowledge of system model, as long as the number of experts units is sufficient. This is because the basic building block of the WTAE is a small single hidden layer MLP network, which realize the nonlinearity of the system locally. This is a great advantage of a neural networks based algorithm over the traditional knowledge based tracking system. WTAE networks have outperformed other neural network based systems in complex problems. Based on the 'divide and conquer' strategy, the model employs a growing fuzzy clustering algorithm that naturally divides the input space into overlapping regions on which 'experts' act. In this way, a complicated problem is divided into a series of simpler sub-problems and assigned a function approximator to each sub-problem locally.

The growing fuzzy clustering algorithm is able to make soft decision boundaries for overlapping classes. This overlapping is another feature of the input sample sets in sensor tracking problems. The algorithm we used overcomes the overlapping problems by using Gaussian fuzzy membership functions in each dimension of the input space. A comparison performance among the three algorithms is listed in Table 1.

A time window was defined with a length of 300 points. Both the real sensor time series and the tracking time series were transferred to the frequency domain by FFT with a length of 256. Both the power spectrum density and the autocorrelation of the residual were calculated. The necessary information for establishing the three fuzzy sensor validation gates was recorded.

Compared with the traditional time domain indicators, WTAE network is more reliable and robust in a noisy environment, as shown in Table 2. When the amplitude of the noise is not significant enough to trigger the noise indicator, the traditional indicators could not detect several failure modes, especially 'Cyclic', which means in

Table 1
A comparison performance among estimators

41 mins training	WTAE networks	MLP network	RBF network
Architecture	23 of 10 hidden nodes MLP experts	150 hidden nodes	339 clusters in the first layer
Training algorithm	Growing fuzzy resilient-BP	Resilient-BP	*k*-Means clustering Moore–Penrose
Training samples	3000	3000	775
MSE of training	0.0234	0.0554	0.0550
Testing samples	3000	3000	775
MSE of testing	0.1631	5.9529	0.0787

Table 2
Comparison results of three validation gates and fuzzy intersection output

Sensor states	Sensor value validation gate	Residual autocorrelation validation gate		Power spectrum validation gate		Fuzzy intersection
		Manhattan distance	Confidence level	Manhattan distance	Confidence level	
Normal	0.9046	1.6723e+003	0.9832	0.0771	0.9447	0.9046
Hard-over	2.8921e−014	6.1907e+005	1.0613e−008	2.7423	1.0854e−024	1.0854e−024
Bias	7.1865e−016	7.2108e+005	1.0613e−008	3.1193	1.0854e−024	1.0854e−024
Spike	7.1858e−016	1.6460e+003	0.9989	0.0822	0.9721	7.1858e−016
Stuck	0.8460	5.6381e+003	1.0591e−008	1.5572	1.0854e−024	1.0854e−024
Erratic	0.9268	1.5550e+004	1.0613e−008	0.4496	1.0854e−024	1.0854e−024
Cyclic	0.4501	4.4328e+004	1.0613e−008	0.4698	1.0854e−024	1.0854e−024
Drift	1.2242e−006	6.1678e+004	1.0613e−008	0.8432	1.0854e−024	1.0854e−024
Nonlinear	5.2260e−014	3.1871e+004	1.0613e−008	1.1034	1.0854e−024	1.0854e−024
Correction Rate	66.67%	88.89%		88.89%		100%

Table 3
A comparison performance among estimators

63 Mins Training	WTAE networks	MLP network	RBF network
Architecture	35 of 15 hidden nodes MLP experts	150 hidden nodes	385 clusters in the first layer
Training algorithm	Growing fuzzy	Resilient BP resilient BP	*k*-Means clustering Moore–Penrose
Training samples	3000	3000	813
MSE of training	0.3985	0.7756	42.5571
Testing samples	3000	3000	813
MSE of testing	2.1402	45.9567	42.3122

the same condition the traditional indicators could only detect less than 87.5% of the available failure states. In addition, the traditional indicators need sufficient knowledge of the characteristics of both the sensor and the environment, which is unnecessary for the proposed WTAE network.

3.2. The Westland vibration data set

The Westland data set [19] was acquired using an array of eight accelerometers fixed in specific locations on a set of faulted and unfaulted aft main power transmission of a US Navy CH-46 helicopter. These accelerometer-equipped transmissions were mounted on a laboratory-based "test rig" and run at a sampling rate of 103,116.08 Hz. The sensor validation experiment data set was sampled at a no-fault condition at one of the several torque load levels (i.e. 100%).

Because of the complexity of the data set, we included 3000 samples both in the training set and testing set. The estimation results were listed in Table 3.

Although we added more hidden neurons to the experts than the first application, the MSE of both training and testing sets are still significant. The other two networks also share the similar results as WTAE network. The possible reason is that the sensors used are probably not strongly correlated.

4. Conclusions

An architecture for estimating sensor measurements and detecting sensor failure is developed in this paper. The method allows us to estimate a critical sensor value when other neighboring sensors measurements are used as inputs. Three fuzzy sensor validation gates based on the information from both the time domain and the frequency domain were used to detect the sensor failure. The network is a synergetic combination of fuzzy logic and neural network. It employs both the fast parallel computation and learning capability of neural networks, and fuzzy logic's ability to represent and manipulate imprecise information.

The WTAE network consists of two main layers: the fuzzy membership clustering layer and the MLP experts layer. The cluster layer employs the Gaussian radial basis function as a fuzzy membership function. The general idea is to divide a complicated problem into a series of simpler sub-problems and assign a function approximator to each sub-problem. A growing fuzzy membership clustering method is used to divide the input space into overlapping regions on which 'experts' act. After the WTAE network was fully trained in the sensor nominal state, the estimation result was compared with real sensor outputs by three fuzzy validation gates, which were built up based on the information collected in the learning stage. The auto-correlation of the residual and the power spectrum of the time series are analyzed using the Manhattan distance. The results from the three validation gates were combined together by fuzzy intersection logic. This decision level fusion finishes the whole sensor failure detection procedure providing the final confidence level of the interested sensor health.

Two benchmark data sets, the SQ Machinery Fault Simulator data set and the Westland vibration data set, were used in simulation studies to demonstrate the performance of the WTAE network. Comparisons between the WTAE networks and the other two neural networks estimators were made. The results show that, in terms of estimation performance, the WTAE is competitive with or even superior to the MLP and RBF networks. Furthermore, the fuzzy sensor validation gates algorithm was used to investigate the eight sensor failure modes. The results from the simulation studies have shown that the proposed sensor validation algorithm is capable of detecting all eight faults as long as the basic assumption that the neighboring sensors have an analytical relationship is valid.

Acknowledgements

This work was supported by the US Air Force Office of Scientific Research under Grant F49620-98-1-0049 and US Environmental Protection Agency under Grant R826273-01-0.

References

[1] G. Yen, Health monitoring on vibration signatures in rotorcraft wings, Neural Processing Letter 4 (3) (1996) 127–137.

[2] P. Meesad, G. Yen, Pattern classification by a neurofuzzy network: application to vibration monitoring, ISA Transactions 39 (3) (2000) 293–308.

[3] C. Smith, G. Erickson, Multisensor data fusion: concepts and principles, IEEE Communications, Computers and Signal Processing 1 (1991) 235–237.

[4] S. Lee, Sensor value validation based on systematic exploration of the sensor redundancy for fault diagnosis KBS, IEEE Transactions on Systems, Man, and Cybernetics 24 (4) (1994) 594–605.

[5] K. Goebel, Management of Uncertainty in Sensor Validation, Sensor Fusion, and Diagnosis of Mechanical System Using Soft Computing Techniques. PhD dissertation, University of California at Berkeley, 1996.

[6] T. Brownell, Neural networks for sensor management and diagnostics, in: Proceedings of the IEEE International Conference on Neural Networks, 1992, pp. 923–929.

[7] E. Eryurek, B. Upadhyaya, Sensor validation for power plants using adaptive back-propagation neural network, IEEE Transactions on Nuclear Science 37 (2) (1990) 1040–1047.

[8] T. Guo, J. Nurre, Sensor failure detection and recovery by neural networks, in: Proceedings of the International Joint Conference on Neural Networks, 1991, pp. 221–226.

[9] M. Napolitano, G. Silverstri, Sensor validation using hardware-based on-line learning neural networks, IEEE Transactions on Aerospace and Electronics Systems 34 (2) (1998) 456–468.

[10] K. Wheeler, A. Dhawan, SSME parameter estimation using radial basis function neural networks, in: Proceedings of the IEEE International Conference on Neural Networks, 1994, pp. 3352–3357.

[11] B. Chandrasekaran, W. Punch, Hierarchical classification: its usefulness for diagnosis and sensor validation, IEEE Journal on Selected Areas in Communications 6 (5) (1999) 884–891.

[12] M. Jordan, R. Jacob, Hierarchical mixture of experts and the EM algorithm, Neural Computation 6 (1994) 181–214.

[13] L. Feldkamp, G. Puskorius, A signal processing framework based on dynamic neural networks with application to problems in adaptation, filtering, and classification, Proceeding of the IEEE 86 (11) (1998) 2259–2277.

[14] E. Wang, W. Lou, Designing a soft sensor for a distillation column with the fuzzy distributed radial basis function neural network, in: Proceedings of the IEEE Conference on Decision and Control, 1996, pp. 1714–1719.

[15] G. Yen, K. Lin, Wavelet packet feature extraction for vibration monitoring, IEEE Transactions on Industrial Electronics 47 (3) (2000) 650–667.

[16] S. Yung, Signal Processing in Local Sensor Validation, PhD Dissertation, University of Oxford, 1992.

[17] V. Vapnik, A. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities, Theory of Probability and its Applications 16 (2) (1971) 264–280.

[18] E. Baum, D. Haussler, What size net gives valid generalization? Neural Computation 1 (1989) 151–160.

[19] B. Cameron, Final report on CH-46 aft transmission seeded fault testing, Westland Research Paper RP907, Westland Helicopters, Ltd., 1993.

[20] R. Wheeden, A. Zygmund, Measure and Integral, Marcel Dekker, New York, 1974.

Gary G. Yen received the PhD degree in electrical and computer engineering from the University of Notre Dame, Notre Dame, IN in 1992. He is currently an Associate Professor in the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK. Before he joined OSU in 1997, he was with the Structure Control Division, US Air Force Research Laboratory, Albuquerque, NM. His research is supported by the DoD, DoE, EPA, NASA, NSF and Process Industry. His research interest includes intelligent control, computational intelligence, conditional health monitoring, signal processing and their industrial/defense applications. Dr. Yen serves as ISA Review Chair for American Control Conference since 1998. Dr. Yen is a senior member of ISA and IEEE.

Wei Feng was born in Beijing, China, in 1974. He received the BS degree in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 1996, and the MS degree in electrical engineering from Oklahoma, State University, OK, USA, in 2000. He joined the Intelligent System and Control Laboratories, Department of Electrical and Computer Engineering, Oklahoma State University, USA, in 1998. He was engaged in research on development of intelligent system for sensor fusion and sensor validation.

APPENDIX C:

Pattern Classification by a Neurofuzzy Network: Application to Vibration Monitoring

by

Phayung Meesad and Gary G. Yen

ISA Transactions, **39**(3), 2000, pp. 293-308

ELSEVIER

ISA Transactions 39 (2000) 293–308

ISA TRANSACTIONS®

www.elsevier.com/locate/isatrans

Pattern classification by a neurofuzzy network: application to vibration monitoring

Phayung Meesad, Gary G. Yen *

Intelligent Systems and Control Laboratory, School of Electrical and Computer Engineering, Oklahoma State University, 202 Engineering South, Stillwater, OK 74078, USA

Abstract

An innovative neurofuzzy network is proposed herein for pattern classification applications, specifically for vibration monitoring. A fuzzy set interpretation is incorporated into the network design to handle imprecise information. A neural network architecture is used to automatically deduce fuzzy if-then rules based on a hybrid supervised learning scheme. The neurofuzzy classifier proposed is equipped with a one-pass, on-line, and incremental learning algorithm. This network can be considered a self-organized classifier with the ability to adaptively learn new information without forgetting old knowledge. The classification performance of the proposed neurofuzzy network is validated on the Fisher's Iris data, which is a well-known benchmark data set. For the generalization capability, the neurofuzzy network can achieve 97.33% correct classification. In addition, to demonstrate the efficiency and effectiveness of the proposed neurofuzzy paradigm, numerical simulations have been performed using the Westland data set. The Westland data set consists of vibration data collected from a US Navy CH-46E helicopter test stand. Using a simple fast Fourier transform technique for feature extraction, the proposed neurofuzzy network has shown promising results. Using various torque levels for training and testing, the network achieved 100% correct classification. © 2000 Elsevier Science Ltd. All rights reserved.

Keywords: Pattern classification; Neural networks; Fuzzy logic; Neurofuzzy classifier; Supervised-clustering learning; Membership functions; Rule bases; Hybrid architecture; Radial basis function

1. Introduction

In machine health monitoring systems, pattern classification is a key component in identifying failure modes created by the monitored systems. Service and maintenance can be promptly and correctly performed if the pattern classifier makes an accurate recommendation. While operating, mechanical components exhibit some physical behaviors, such as temperature, pressure, electromagnetic variation, eddy current, acoustic emission, and vibration, which contain information about the state of the machine. These physical behaviors are sensed by transducer systems to obtain the data used for detecting and diagnosing some of the incipient failures of the machinery and equipment. The input data is entered into a classifier which is a component of a health monitoring system. Using pattern classification techniques, signatures containing information about machine defects and their causes can be extracted from the data. With the accurate decision of a classifier in a

* Corresponding author. Tel.: +1-405-744-7743; fax: +1-405-744-9198.

E-mail address: gyen@ceat.okstate.edu (G.G. Yen), meesad@okstate.edu (P. Meesad).

PII: S0019-0578(00)00027-6

monitoring system, machine maintenance can be performed before catastrophic failures occur.

When a machine is operating properly, the physical behaviors, for example vibrations, are generally small and constant. However, when faults develop which lead to variations of process dynamics, the physical signatures (i.e. power spectrum density, natural frequency, and mode shape) also change. To detect these changes, classical off-line iterative learning classifiers were proposed to supervise a monitored system. These classifiers have a drawback in that they generally require a long training time. In addition, they are often stuck at local minima and are unable to achieve the optimum solution.

Furthermore, in an operating mode, it is possible that novel faults are evolving while a monitored system is running. These faults are different from those that have been trained to the classifier and need to be promptly detected and distinguished from those that have been trained to the classifier. Conventional neural classifiers need to be retrained by both old and new data in order to learn new information while remembering existing information [1]. It is possible to learn and detect new fault patterns on-line real-time if an incremental learning classifier is used. The classifier developed in this study provides the feature of learning new fault types incrementally without re-learn the existing patterns that have been learned. Thus, the proposed is easily applied in a monitoring system to detect unseen faults while the system is in an operating mode.

2. Literature reviews

Pattern classification forms a fundamental solution to problems in real world applications. The function of pattern classification is to categorize an unknown pattern into a distinct class based upon a suitable similarity measure. Thus, similar patterns are placed in the same class while dissimilar patterns are classified into different classes.

Engineers and scientists have developed various methodologies to deal with pattern classification problems. Statistical pattern classification is a traditional technique for classification problems [2]. This classical classification technique makes use of statistical decision theory to classify patterns. Various researchers have scrutinized parametric Bayesian classifiers [3] assuming that the forms of input distributions are known. The parameters of distributions are computed using all training data. The training data is usually assumed to be Gaussian when using Bayesian classifiers. Because of their simplicity, they are still widely used [4,5].

Automatic pattern classification has been actively pursued by scientists and engineers from different fields. Many researchers in the area of pattern classification have paid attention to neural network classifiers because of the capability of model-free and trainable systems, parallel computation, and noise tolerance. These properties of artificial neural networks inspire researchers to study neural network applications to deal with pattern classification problems. Neural networks with the abilities of real-time learning, parallel computation, and self-organization make pattern classification more suitable to handle complex classification problems through their learning and generalization abilities [6,7].

In addition, fuzzy set theory [8] has been extensively applied to pattern classification.. Fuzzy set theory supports pattern classification by dealing with inexact rather than exact notions. Fuzzy systems perform well on uncertain information, very similar to the way human reasoning does. In the real world, most situations are fuzzy rather than crisp. Moreover, the information in pattern classification problems is imprecise rather than precise in nature, and fuzzy set theory allows us to properly model this vague information [9,10].

The integration of neural networks and fuzzy sets is also an active area for pattern classification problems. A growing number of researchers have designed and examined various forms of fuzzy neural or neurofuzzy networks. The idea is to integrate the capabilities of model-free and trainable systems, parallel computation, and noise tolerance of neural networks and the ability of fuzzy set theory to deal with imprecise situations. The combination of neural networks and fuzzy sets forms a synergetic network that handles pattern classification problems very effectively and efficiently. Some learning algorithms of fuzzy neural or neurofuzzy networks can be found from [1,11–14]. Fuzzy neural or neurofuzzy networks have been widely used in many applications in pattern classification problems shown in [11–17].

2.1. *Existing problems*

Pattern classification techniques have become important in handling many real-world applications. As a complement to statistic classifiers, neural network classifiers, fuzzy classifiers, and neural-fuzzy classifiers have been applied to deal with classification problems. However, those neural network, fuzzy, and neural-fuzzy classifiers have some deficiencies in many aspects.

For example, the mutilayer percerotron (MLP) and the learning vector quantization (LVQ) classifiers require off-line training and iterative presentation of the training data. Thus, they use extensive training time to learn input patterns. Furthermore, these networks need predetermination of their architecture parameters. Repeated design work is needed to obtain the suitable parameters to achieve reasonable performance in classifying patterns. Moreover, sometimes they fail in the learning process by being unable to converge to the optimal solution because the initial random condition is not properly chosen. Hence, MLP and LVQ classifiers are difficult to apply to pattern classification problems that need fast, on-line, real-time, incremental learning.

Unlike neural net classifiers, fuzzy classifiers can handle uncertain information by providing a soft decision that allows a pattern to belong to more than one class with different membership degrees. However, constructing "if–then" rules for fuzzy classifiers requires knowledge from experts and a time-consuming design process, especially when the dimension of the feature space is large. Thus, traditional fuzzy classifiers are not suitable for on-line, real-time, incremental learning pattern classification.

The aim of this study is to provide a way for determining fuzzy if–then rules during the learning phase. The proposed method can automatically construct the if–then rules of a fuzzy inference system using the learning capability of a neural network. Input and target pairs are used as a "teacher" for the network under supervised learning. This produces an incremental. on-line, one-pass learning algorithm. This model uses Gaussian membership functions in both the antecedent and consequent parts. The proposed method employs supervised clustering-based partitioning.

3. Architecture of the proposed neurofuzzy network

The proposed neurofuzzy network is developed based on a standard fuzzy logic system [18]. A standard fuzzy logic system has four components: a fuzzifier, a fuzzy rule base, an inference engine, and a deffuzifier. The function of the fuzzifier is to determine the degree of membership of a crisp input in a fuzzy set. The fuzzy rule base is used to represent the fuzzy relationships between input and output fuzzy variables. The output of the fuzzy rule base is determined based on the degree of membership specified by the fuzzifier. The inference engine controls the rule base. Optionally, the defuzzifier is used to convert the outputs of the fuzzy rule base into crisp values. The network architecture of the proposed neurofuzzy classifier is shown in Fig. 1.

The proposed neurofuzzy system has three layers: one input layer, one hidden or rule layer, and one output layer. In the input layer, each neuron connects to each element of an M-dimensional input vector. The hidden layer or rule layer separates into two parts: an antecedent part and a consequent part. The antecedent and consequent together construct a fuzzy rule for a fuzzy inference system. For each rule, the antecedent part consists of M membership nodes and a fuzzy "AND" activation function node. Gaussian membership functions are employed at the membership nodes. Each membership node of the antecedent part orderly connects to each node of the M-dimensional input layer via a dynamic synaptic weight matrix, $\mathbf{W_P}$, whose rows represent prototype vectors which are the centroids of Gaussian radial basis functions. When adding more rules in the hidden layer, $\mathbf{W_P}$ adds more rows. $\mathbf{W_P}$ is a long-term-memory trainable weight. In the antecedent parts, a fuzzification procedure and a fuzzy "AND" operation are performed.

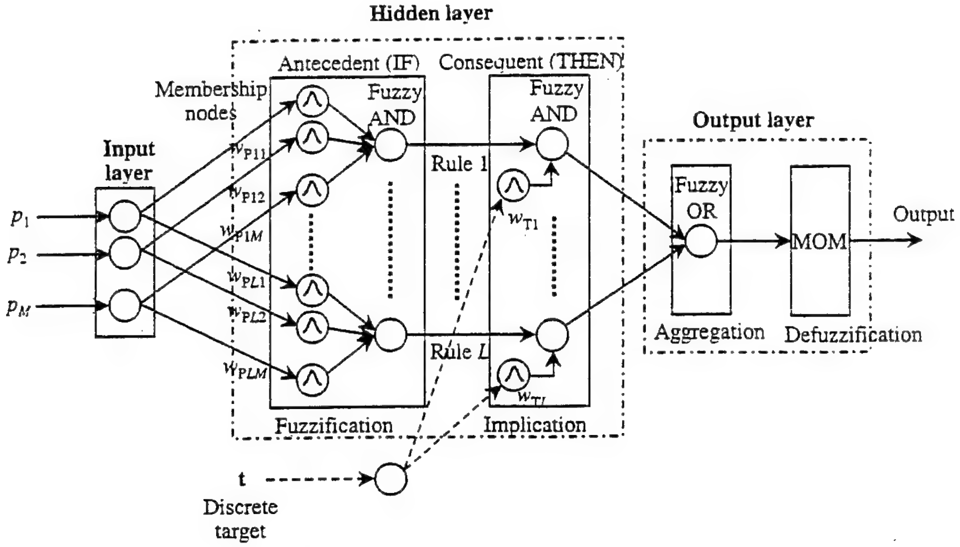


Fig. 1. The proposed neurofuzzy network architecture.

The consequent part consists of target membership functions and a fuzzy “AND” activation function. Consequent membership functions are Gaussian membership functions (centered by the elements of $\mathbf{W_T}$) as in the antecedent part. The consequent of a fuzzy rule assigns an entire fuzzy set to the output. This fuzzy set is represented by a membership function that is selected to show the qualities of the consequent. If the antecedent is only partially true, (i.e. is taken a value less than “1”), then the output fuzzy set is truncated which is often referred to as implication method [18]. After implication procedure, the fuzzy numerical output of the rule layer is then transmitted to the output layer.

The output layer is composed of an aggregation procedure and a defuzzification procedure. The aggregation step combines the inference results of fuzzy rules from the hidden layer by superimposing all fuzzy conclusions about a variable. The aggregation procedure employs a fuzzy “OR” (maximum) method. For the defuzzification step, the mean of maximum method (MOM) is used to obtain the final crisp output. The MOM defuzzification calculates the average of all output's variable values with maximum membership degrees.

3.1. Mathematical model of the neurofuzzy classifier

Let $\mathcal{R}^M$ be a pattern vector space. Let $\mathbf{p} = [p_1 p_2 \dots p_M]^T \in \mathcal{R}^M$ be an input pattern. Each element of the input pattern is a measurement or feature and each corresponds to one dimension (axis) in the space. For M elements of the input pattern we have an M-dimensional space, or M-space. Let vectors $\mathbf{w}_{Pi} = [w_{Pi1} w_{Pi2} \dots w_{PiM}]^T$, $i = 1, \dots, L$, be pattern prototypes of the pattern space. L denotes the number of pattern prototypes. Let matrix $\mathbf{W_P} = [\mathbf{w}_{P1} \mathbf{w}_{P2} \dots \mathbf{w}_{PL}]^T$ be a synaptic weight matrix whose row vectors represent prototypes of the pattern space. Pattern prototypes are the centers of the Gaussian membership functions of the antecedent part in the rule layer. A class may have more than one prototype. Each prototype $\mathbf{w}_{Pi}$, $i = 1, \dots, L$, is the mean vector of the patterns that belong to the ith rule. The antecedent part of a rule is constructed from a pattern prototype. Each rule consists of M membership nodes. Let $\mathcal{R}^1$ be a one-dimensional target space. Let $\mathbf{t} \in \mathcal{R}^1$ be a corresponding target or class of the input pattern $\mathbf{p}$. Let vector $\mathbf{W_T} = [w_{T1} w_{T2} \dots w_{TL}]^T$ be a target vector whose each element, w_{Ti}, $i = 1, \ldots, L$, represents a target of a prototype stored in $\mathbf{W_P}$ in the same order as their neurons. Each element of $\mathbf{W_T}$ is a center of a Gaussian membership function of the consequent part in the rule layer. The number of rows of $\mathbf{W_P}$ and $\mathbf{W_T}$ are the same and they grow dynamically as more rules are added to the hidden layer.

3.2. Gaussian membership function and similarity function

Gaussian membership functions are employed in the hidden layer to represent the degree of similarity between the input pattern and the reference prototypes. For an M-dimensional pattern space, M membership functions are used to form a prototype. These Gaussian membership functions form the antecedent part of a fuzzy rule. A Gaussian membership function is computed by the following equation:

$$\mu(p_m, w_{\mathrm{P}im}) = \exp\left[-\left(\frac{w_{\mathrm{P}im} - p_m}{\sqrt{2}\sigma_{im}}\right)^2\right], \quad m = 1, \ldots, M; i = 1, \ldots, L, \tag{1}$$

where $\mu(p_m, w_{\mathrm{P}im})$ is a membership degree of p_m that belongs to prototype $\mathbf{w}_{\mathbf{P}i}$ in the mth dimension; p_m is the mth element of an M-dimensional input pattern; $w_{\mathrm{P}im}$ represents the center of a Gaussian membership function of prototype $\mathbf{w}_{\mathbf{P}i}$ in the mth dimension; σ_{im} is the standard deviation of a Gaussian function at the mth dimension of the ith prototype, i.e. the standard deviation of ith prototype $\sigma_i = [\sigma_{i1}, \sigma_{i2}, \ldots, \sigma_{iM}]$, $i = 1, \ldots, L$; and L is the number of prototypes in the pattern space.

A degree of similarity between the input pattern $\mathbf{p}$ and the reference prototype $\mathbf{w}_{\mathbf{P}i}$ is called "similarity function" determined as the following equation:

$$\mu_S(\mathbf{p}, \mathbf{w}_{\mathbf{P}i}) = \min_m(\mu(p_m, w_{\mathrm{P}im})), i = 1, \ldots, L. \tag{2}$$

A prototype with highest membership degree is called a "winner prototype" which can be found by the following equation:

$$\mathbf{w}_{\mathbf{P}J} = \left\{\mathbf{w}_{\mathbf{P}i} \middle| \max_i(\mu_S(\mathbf{p}, \mathbf{w}_{\mathbf{P}i}) \times CF_i)\right\}, \tag{3}$$

where $J \in \{1, \ldots, L\}$ is the winner's index; and CF_i is the confident factor of ith prototype. CF_i, which will be discussed later, is determined by taking account for the frequency of patterns that fall into the region of the ith prototype with respect to all available patterns in the training set. The term fuzzy "AND" may be alternatively used instead of the operator "min." The term fuzzy "OR" and the operator "max" may be used interchangeably.

The similarity function, $\mu_S(\mathbf{p}, \mathbf{w}_{\mathbf{P}i})$, is defined such that for all patterns, $\mathbf{p}$'s, within the region describing $\mathbf{w}_{\mathbf{P}i}$, $i = 1, \ldots, L$, there exists a function $\mu_S(\mathbf{p}, \mathbf{w}_{\mathbf{P}i})$ such that $\mu_S(\mathbf{p}, \mathbf{w}_{\mathbf{P}i}) > \mu_S(\mathbf{p}, \mathbf{w}_{\mathbf{P}j})$ for all $j \neq i$. Fig. 2a illustrates an example of a one-dimensional pattern space with three pattern prototypes: $\mathbf{w}_{\mathbf{P}1}$, $\mathbf{w}_{\mathbf{P}2}$, $\mathbf{w}_{\mathbf{P}3}$. Since this is a one-dimensional pattern space, only one membership function is used for each prototype. Fig. 2a is an example of point $\mathbf{p}$ classified to be the class of $\mathbf{w}_{\mathbf{P}2}$ since $\mu_S(\mathbf{p}, \mathbf{w}_{\mathbf{P}2})$ is larger than both $\mu_S(\mathbf{p}, \mathbf{w}_{\mathbf{P},1})$ and $\mu_S(\mathbf{p}, \mathbf{w}_{\mathbf{P},3})$.

Fig. 2b shows an example of a two-dimensional pattern space with 3 prototypes. For two-dimensional pattern space, each prototype is formed using two membership functions. Applying Eqs. (1)–(3), by inspection, pattern $\mathbf{p}$ in Fig. 2b is determined to be the class of prototype $\mathbf{w}_{\mathbf{P}1}$ since $\mu_S(\mathbf{p}, \mathbf{w}_{\mathbf{P}1})$ is larger than both $\mu_S(\mathbf{p}, \mathbf{w}_{\mathbf{P}2})$ and $\mu_S(\mathbf{p}, \mathbf{w}_{\mathbf{P}3})$. The above idea applies, in similar fashion, to an arbitrary M-dimensional pattern space with L pattern prototypes.

3.3. Neurofuzzy classifier learning scheme

In the neurofuzzy network, there are two cases of learning: learning of new prototypes and learning of existing prototypes. In the first case, learning of new prototypes is the way that a new prototype is added in the hidden layer, i.e. $\mathbf{W_P}$ and $\mathbf{W_T}$ grow new neurons. This case is conducted when a new prototype is found. $\mathbf{W_P}$ grows by adding a current input pattern $\mathbf{p}$ to its structure. Pattern $\mathbf{p}$ forms a new prototype for the pattern space, i.e. pattern $\mathbf{p}$ is the new centers of Gaussian membership functions in the antecedent part of

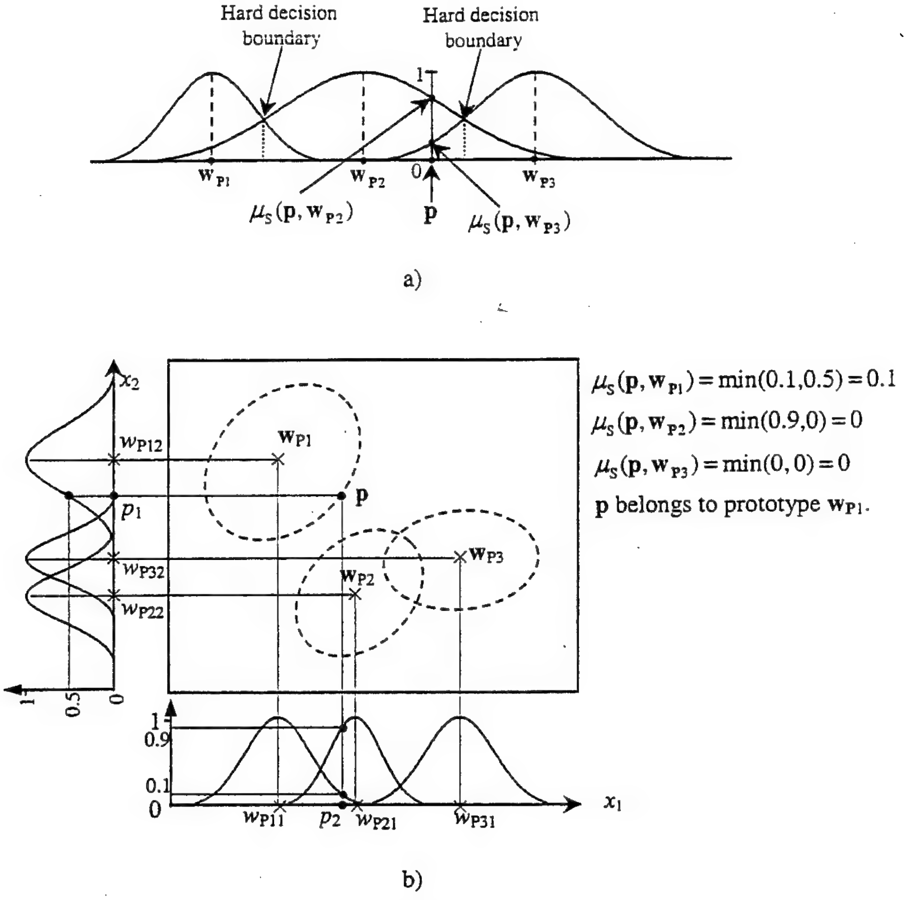

Hard decision boundary
Hard decision boundary
1
0
$\mathbf{w}_{P1}$
$\mathbf{w}_{P2}$
$\mathbf{w}_{P3}$
$\mu_S(\mathbf{p},\mathbf{w}_{P2})$
$\mathbf{p}$
$\mu_S(\mathbf{p},\mathbf{w}_{P3})$
a)
x_2
w_{P12}
p_1
w_{P32}
w_{P22}
$\mathbf{w}_{P1}$
$\mathbf{p}$
$\mathbf{w}_{P2}$
$\mathbf{w}_{P3}$
1
0.5
0
1
0.9
0.1
0
w_{P11}
p_2
w_{P21}
w_{P31}
x_1
$\mu_S(\mathbf{p},\mathbf{w}_{P1}) = \min(0.1, 0.5) = 0.1$
$\mu_S(\mathbf{p},\mathbf{w}_{P2}) = \min(0.9, 0) = 0$
$\mu_S(\mathbf{p},\mathbf{w}_{P3}) = \min(0, 0) = 0$
$\mathbf{p}$ belongs to prototype $\mathbf{w}_{P1}$.
b)

Fig. 2. (a) One-dimensional pattern space; (b) two-dimensional pattern space.

the rule layer. Similarly, $\mathbf{W}_T$ grows a new neuron when $\mathbf{W}_P$ grows. A target label of the pattern $\mathbf{p}$ is added to $\mathbf{W}_T$ and it is constructed to be a new center of the consequent Gaussian membership function.

The second case, learning of existing prototypes, is applied when a pattern $\mathbf{p}$ is considered to have the same target as an existing prototype. This learning process adapts the shape of existing membership functions, i.e. means and standard deviations of Gaussian membership functions. The learning of existing prototype of the neurofuzzy network takes place in the antecedent part of the hidden (rule) layer. (The learning of existing prototypes does not take place in the consequent part since the means and standard deviations of membership function in the consequent part are fixed.)

In the learning of an existing prototype case, the neurofuzzy network employs a supervised-clustering learning paradigm that clusters input data using the corresponding targets to validate group prototypes. A presented input pattern is measured using the similarity with existing cluster prototypes. If the criterion of similarity is met, it is included into the selected prototype provided that it has the same target. Once an input pattern is included in a prototype, only the parameters, i.e. count, sum of square, mean, and standard deviation, of the selected prototype are updated as follows:

$$C_J^{(n+1)} = C_J^{(n)} + 1, \tag{4}$$

$$\mathbf{ssq}_J^{(n+1)} = \mathbf{ssq}_J^{(n)} + \mathbf{p}^2, \tag{5}$$

$$\mathrm{w}_{\mathbf{P}J}^{(n+1)} = \frac{\mathbf{w}_{\mathbf{P}J}^{(n)}\left(\mathrm{C}_J^{(n+1)} - 1\right) + \mathbf{p}}{\mathrm{C}_J^{(n+1)}}, \tag{6}$$

$$\sigma_J^{(n+1)}) = \begin{cases} \sigma_0 & \text{for } \mathrm{C}_J = 1, \\ \sqrt{\dfrac{\mathbf{ssq}_J^{(n+1)} - \mathrm{C}_J^{(n+1)}\left(\mathbf{w}_{\mathbf{P}J}^{(n+1)}\right)^2}{\mathrm{C}_J^{(n+1)} - 1}} & \text{otherwise.} \end{cases} \tag{7}$$

n represents a time index. C_J represents the number of inputs that have been counted into the Jth prototype. Each element of $\mathbf{w}_{\mathbf{P}J}$ is the center of a Gaussian membership function of the Jth prototype. $\mathbf{ssq}_J$ is the sum of square of patterns that have been included to the Jth prototype. The standard deviation, σ_J, will be used to indicate the spread of the data in the Jth prototype. σ_0 is the initial standard deviations representing the isotropic spread in pattern space of a new prototype for the first sample.

3.4. Fuzzy if-then rule bases

In standard fuzzy system, for a finite class pattern classification problem with an M-dimensional pattern space, linguistic knowledge can be written as a set of fuzzy if-then rule as follows:

$$R_i: \quad \mathrm{IF} p_1 \,\mathrm{is}\, A_{i1} \,\mathrm{AND} \ldots \mathrm{AND} p_M \,\mathrm{is}\, A_{im}$$
$$\mathrm{THEN\,Class} T_i \,\mathrm{with\,confident\,factor} = CF_i$$

where R_i, $i = 1, 2 \ldots, L$, is the label of the ith rule, A_{im}, $m = 1, 1, \ldots, M$; is the antecedent fuzzy set of the ith rule for the mth dimension; T_i is the consequent class; CF_i is the grade of certainty that the input pattern belongs to class T_i; and L is the number of fuzzy rules. Each antecedent fuzzy set A_{im} has a linguistic label such as "small," "medium," or "large." The grade of certainty CF_i is usually given as a real number in the unit interval "0" and "1."

From a knowledge representation viewpoint, it is preferable to represent fuzzy rules in linguistic form as in the standard fuzzy system. For the sake of fast, one-pass, learning behavior, the neurofuzzy classifier does not employ fuzzy linguistic rules. However, a mapping algorithm can be used to map fuzzy if–then rules of the neurofuzzy classifier into fuzzy linguistic form that is a more comprehensible form for human users.

Fuzzy if–then rules of the neurofuzzy classifier automatically constructed can be interpreted as follows:

- Rule 1: IF (p_1 is A_{11}) AND (p_2 is A_{12}) AND ... AND (p_M is A_{1M}) THEN Class is T_1 with CF_1;
- Rule 2: IF (p_1 is A_{21}) AND (p_2 is A_{22}) AND ... AND (p_M is A_{2M}) THEN Class is T_2 with CF_2;

  ...
- Rule L: IF (p_1 is A_{L1}) AND (p_2 is A_{L2}) AND ... AND (p_M is A_{LM}) THEN Class is T_L with CF_L; where p_m, $m = 1, \ldots, M$, is the mth element of an input pattern $\mathbf{p}$; A_{im} is an antecedent membership function at node m of the ith rule, $i = 1, \ldots, L$, (an antecedent membership function is a Gaussian radial basis function centered by an element in vector $\mathbf{w}_{\mathbf{P}i}$); T_i, $i = 1, \ldots, L$, represents a consequent membership function of the ith rule (i.e. T_i is the ith consequent membership function which is a Gaussian function centered by the ith element in $\mathbf{W}_\mathbf{T}$); and CF_i, $i = 1, \ldots, L$, is the confident factor of the ith rule. By applying the concept of the probability, CF_i can be determined by the following equation:

$$CF_i = \frac{C_i}{\sum\limits_{j \in \mathrm{Class}(w_{\mathrm{T}i})} C_j}, i = 1, \ldots, L, \tag{8}$$

where C_i is a count parameter of the ith rule (i.e. ith prototype) obtained when a pattern is included into the ith prototype; and $\mathrm{Class}(w_{\mathrm{T}i}) = \{k | w_{\mathrm{T}k} = w_{\mathrm{T}i}, k = 1, \ldots, L\}$.

3.5. Defuzzification method

A defuzzification process is applied, in the output layer, to obtain the final crisp output. The defuzzification method used in this study is defined as the mean of maximum (MOM) by the following equation:

$$\mathrm{MOM} = \frac{\sum_{i=1}^{Q} y_i}{\#\text{ of elements in } \mathbf{y}}, \tag{9}$$

$$\mathbf{y} = \left\{ y \,\middle|\, \mu(y) = \max_{\forall Y}(\mu(Y)) \right\}, \tag{10}$$

where $\mathbf{y} = [y_1 \ldots y_Q]$ is the set of output values with the highest membership degree; Q is the dimension of $\mathbf{y}$; Y is the output "universe of discourse" which is the range of values the outputs can take on; and $\mu(y)$ defines a membership function for y.

3.6. *Neurofuzzy classifier algorithm*

The neurofuzzy network will be automatically constructed on the run by supervised-clustering algorithm. (Supervised-clustering learning distinguishes patterns into sub-clusters by using the corresponding targets of the input patterns to decide whether the data should be included in the same group.) Initially, the network contains no rules in the hidden layer. When the training data is presented, the network learns to construct its weights and connections.

Two parameters needed for the neurofuzzy classifier are the initial standard deviation, σ_0, and the constraint standard deviation, σ_c. When a pattern is included to a cluster, the standard deviation is updated depending on the distribution of patterns in that cluster. However, it is possible that the updated standard deviation becomes very small near zero causing a problem divided by zero. To avoid the divided-by-zero problem, a standard deviation near zero needs to be set to a constraint value larger than zero, for example 0.0001. The classification algorithm of the neurofuzzy classifier is as follows.

- STEP 1: Initialization
  - There are no neurons in the hidden layer.
  - Set count for each node to zero.
  - Set sum of square to zero.
  - Set the initial standard deviation of Gaussian membership function to a small number value. for example 0.1.
  - Set the constraint for standard deviation to a small number value. for example 0.0001.
  - Use the first input pattern to generate the antecedent membership functions and other variables:
    - Set each element of the first input pattern to be the centers or means of the Gaussian functions. i.e. set the mean of each node equal to each element of the input pattern.
    - Set the standard deviation of each node equal to the initial standard deviation.
    - Set the sum of square for each node equal to each element in the input pattern.
  - Construct the first membership function of the consequent part using the first corresponding target of the first input to be the center of the membership function.
- STEP 2: Input presentation
  - Read in the next input pattern (and, optionally, the next corresponding target).
  - Fuzzify each input element using the membership functions in the membership nodes. Each crisp element of the input pattern is fuzzified into membership values by using parameters of corresponding node, i.e. its mean and standard deviation. Each node will have a membership degree for each corresponding input element. (This is in the antecedent part.)
  - Apply fuzzy "AND" (min) operator to membership values from each membership node which yields only one member value from fuzzy "AND" (min) operator.
  - Apply implication method to the membership functions of the consequent part using fuzzy "AND" (min) operator.
  - If there is no presentation of the corresponding target of the input then skip STEP 3.
- STEP 3: Learning step
  - If there is a presentation of the corresponding target of the input (i.e. the system is in training mode), determine which

rule the input most satisfies (i.e. the rule that the input belongs to) by applying fuzzy "OR" (max) operation to the output of the fuzzy "AND" (min) operation.

- If the winner rule meets the criteria of belonging (i.e., its prototype has the highest membership degree and the target of the input pattern is the same as the predicted class), then include that input to the winner rule by recalculating rule parameters, i.e. count, mean, sum of square, and standard deviation.
- If the winner rule does not satisfy the criteria (i.e. the target of the input pattern is not the same as the predicted class) then create a new rule:
    - Set each element of the input pattern to be the centers or means of the Gaussian functions, i.e. set the mean of each node equal to each element of the input pattern;
    - Set the standard deviation of each node equal to the initial standard deviation;
    - Set the sum of square for each node equal to each element in the input pattern;
    - Construct the new membership function of the consequent part using the corresponding target of the present input pattern to be the center of the membership function.

- STEP 4: Final prediction
    - Apply the aggregation method using the fuzzy "OR" operator.
    - To obtain crisp class output, apply the defuzzification method using MOM method.
    - If there is no further input then STOP, otherwise go to **STEP 2**.

4. Simulation results

To demonstrate the performance of the neurofuzzy classifier, software simulations were used in our experiments. The simulation programs were written to run under MATLAB version 5.3 or higher. A Pentium 233MMX PC hosted the simulation programs. Two data sets were used for training and testing the classifier in our studies. The first benchmark data set was the well-known Fisher's Iris data set [19]. Another data set was a time-series vibration data set known as the Westland vibration data set [20]. The details of these experiments are as follows.

4.1. Fisher's Iris flower data set

The Fisher's Iris flower data set consists of 150 patterns and four features: sepal length, sepal width, petal length, and petal width. These four features describe the shape and size of the Iris flowers. Each pattern in the data set falls into one of three classes: Setosa, Versicolour and Virginica, with a total of 50 patterns per class. For the purpose of this experiment, we will call them Class 1, Class 2, and Class 3, respectively. Class 1 is linearly separable from the other two. However, Class 2 and Class 3 are not linearly separable from each other.

Fig. 3a shows the scatter plot of the Iris data for sepal width and length features. It is worth noting from the plot that Class 1 can be easily separated from Class 2 and Class 3. However, Class 2 and Class 3 seem very difficult to separate since there is an overlap between them. Moreover, in Fig. 3b, the petal width and length features are plotted showing that Class 1 is very well separated from Class 2 and Class 3. However, Class 2 and Class 3 remain overlapped [19].

4.1.1. Classification results from the neurofuzzy classifier on Iris data

In this study, 10 trials were performed by randomly selecting the order of the data. For each trial, the training set was composed of the first 75 patterns. The other 75 patterns were used to test the classification performance of the trained network. In this study, the initial standard deviation, σ_0, was set to 0.5. The constraint of the updated standard deviation, σ_c, was set to 0.01. The results of the study are shown in Table 1.

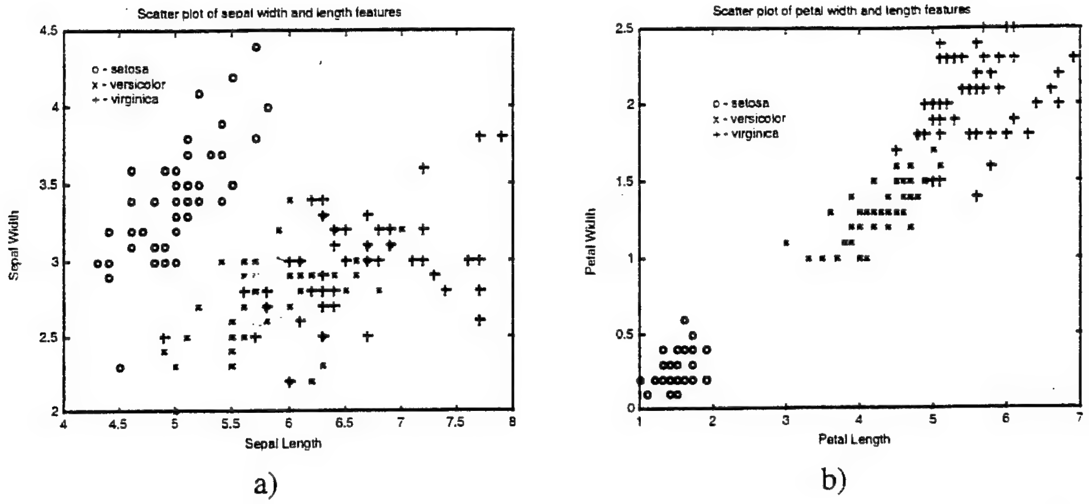


Fig. 3. (a) Scatter plot of sepal width and length features of the Fisher's Iris data; (b) scatter plot of petal width and length features of the Fisher's Iris data.

Table 1
The classification performance of the neurofuzzy classifier on Iris data

No. trials	1	2	3	4	5	6	7	8	9	10
No. rules	18	9	9	11	19	9	11	10	11	11
% correct	97.33	88	97.33	96	90.67	97.33	93.33	92	94.67	96
No. wrong classes	2	9	2	3	7	2	5	6	4	3

Using 75 training patterns, the neurofuzzy classifier performed one pass to automatically generate fuzzy if–then rules (or hidden nodes). From Table 1, the minimum and maximum numbers of rules generated were nine rules and 19 rules, respectively. The minimum correct classification achieved was 88% with nine patterns for wrong classification. A maximum of 97.33% correct classification was achieved in this data set. It is worth noting that neurofuzzy classifier on-line, incrementally learns in only a single pass through all training patterns. The variations of percentages of correct classification are due to the sensitivity of the order of presentation of the training input. By presenting different input orders, the network constructs different rules. In addition, it yields different performances. However, with the maximum percentage of correct classification, the performance of the proposed neurofuzzy classifier for Iris data set was as good as many well-known classifiers (see [1] for references).

4.2. The Westland vibration data set

This data set consists of vibration data recorded using eight accelerometers mounted on different locations on the aft main power transmission of a US Navy CH-46E helicopter. The CH-46E Chinook is a twin-rotor, fore/aft transmission rotorcraft powered by two turbine engines. The data set is archived at the Applied Research Laboratory (ARL) of Penn State University. The Westland vibration data was collected by using an International Recording Instruments Group analog tape recorder and a single mixbox and aft main transmission installed on a test stand and run at nine different torque levels (i.e. 27, 40, 45, 50, 60, 70, 75, 80 and 100%). While collecting the data, only one faulted component was installed in the mixbox and transmission at a time and vibration data was recorded. The data was recorded for seven types of faults and one "no fault," as listed in Table 2. Employing a 10-channel data acquisition

Table 2
A list of the fault types created in the test gearbox

Fault no.	Description
2	Epicyclic planet gear bore/bearing/inner race corrosion spalling
3	Spiral bevel input pinion bearing journal corrosion pitting/spalling
4	Spiral bevel input pinion gear tooth spalling/scuffing
5	High speed helical input pinion tooth chipping and freewheel clutch bearing false brinnelling
6	Helical idler gear crack propogation
7	Collector gear crack propogation
8	Quill shaft crack propogation
9	No defect

system, the data was digitized at a sample rate of 103,116.08 Hz with a 16-bit quantization level and was saved in 1.506 MB data files. All together, there are 71 files with each file containing all eight accelerometer signals. The data files used in this study were 1-s data files [20].

4.2.1. Westland data characteristics

Figs. 4a and b show two samples of vibration data in the time domain pertaining to fault Class 2 and Class 3 from Accelerometer 1 of the Westland Data Archive. However, it is difficult to discriminate the two raw time-series data. The raw time series data provides little information to use for classification. It is preferable to transform the signal from the time domain to the frequency domain by using the fast Fourier transform (FFT) technique. The vibration signatures in frequency domain are shown in Figs. 5a and b, which are power spectral density plots of the two signals given in Figs. 4a and b, respectively. It is easy to see that frequency content above 20 kHz is less useful. The effective information for classification is in the frequency range of 3–10 kHz. For the interested frequency range of 0–12 kHz, Figs. 6a and b illustrate a "zoom-in" version of the power spectrum density plot shown in Figs. 5a and b, respectively.

More sample plots in the frequency domain for 100% torque level of the Westland vibration data set is shown in Fig. 7. Fig. 7 shows sample patterns of faults 2, 3, 4, 5, 6, 7, 8, and "no fault" from all eight accelerometers. It is worth noting that data from each sensor alone is not sufficient to classify all fault classes. Moreover, it is easier to classify the data by using all patterns obtained from all eight sensors. In this study, most of our experiments used the combined signatures from all eight sensors as training patterns.

4.2.2. The performance of the neurofuzzy classifier on the Westland data set

In our experiments, vibration time-series data was preprocessed using the FFT technique to transform from the time domain to the frequency domain. Power spectrum command (SPECTRUM in Matlab Signal Processing Toolbox) with a Hanning window of 1024 samples was utilized. The data was filtered with the interested frequency band of 3–10kHz, obtaining a 141×1 vector for each channel. Vectors from the eight channels were set into one vector (1128×1 vector). A list of

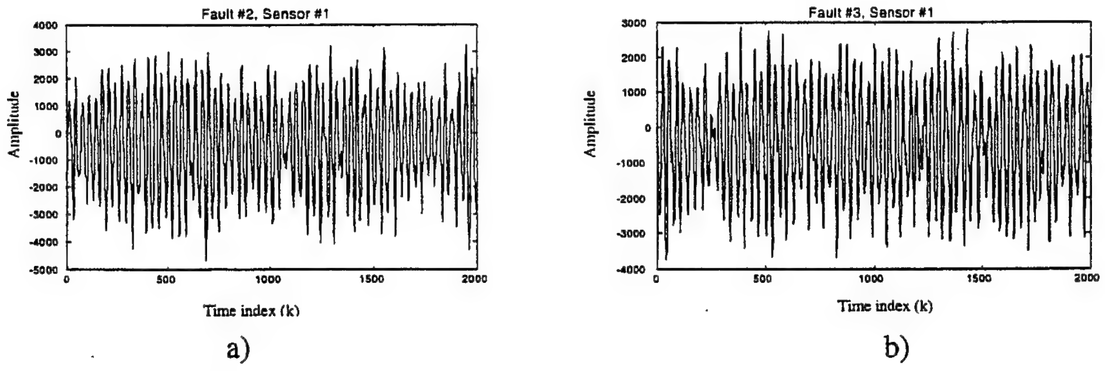


Fig. 4. (a) A plot of time series data of fault 2 from sensor 1; (b) a plot of time series data of fault 3 from sensor 1.

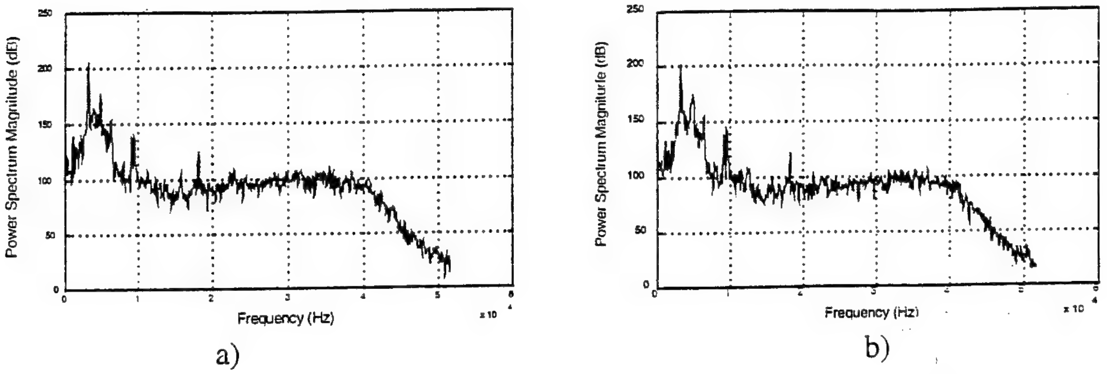


Fig. 5. (a) Power spectrum density db plot of fault 2 from sensor 1; (b) power spectrum density db plot of fault 3 from sensor 1.

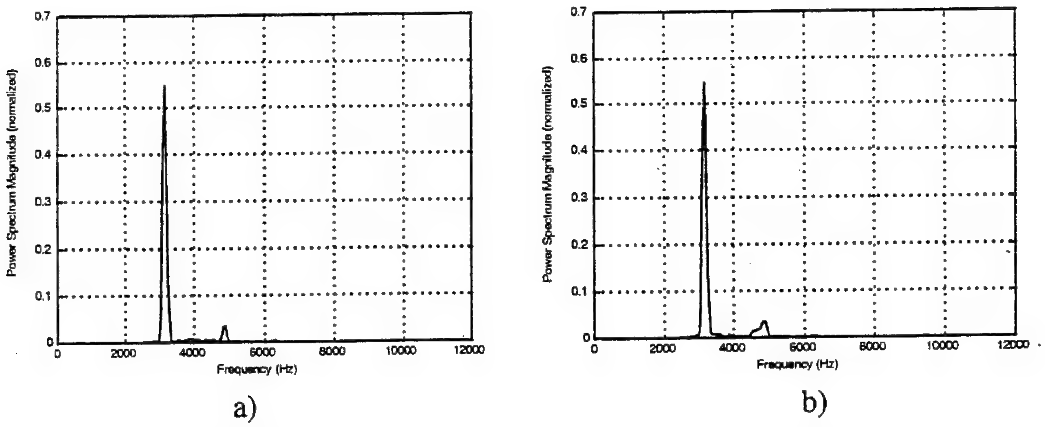


Fig. 6. (a) Power spectrum density plot of fault 2 from sensor 1; (b) power spectrum density plot of fault 3 from sensor 1.

torque levels, fault types, and number of patterns used in this study are shown in Table 3.

In this study, the initial standard deviation, σ_0, was set to 0.5. The constraint of the updated standard deviation, σ_c, was set to 0.01. Table 4 shows the results of the simulation study for the performance of the neurofuzzy classifier on the Westland vibration data. All torque levels (i.e. 27, 40, 45, 50, 60, 70, 75, 80, and 100%) were used both for training and testing to the neurofuzzy classifier. When the same torque level, only 100 patterns were used for training, and the remaining patterns were used for testing. All available patterns were used for training when different torque levels were used for testing. For the last column of Table 4, 100 patterns from every torque level were used for training. Then all available patterns were used for testing.

The complete simulation results of the neurofuzzy classifier on the Westland vibration data are shown in Table 4. In Table 4, the columns represent the "training data" with different torque levels, and the rows indicate the "testing data" with different torque levels. A percentage of correct classification is interpreted by crossing each column with each row. For instance, 100% correct classification was achieved when the neurofuzzy network was trained by the 40% torque level and was tested by the 45% torque level. The numbers of rules resulting from the training process of the fuzzy classifier are shown in Table 4 in the row leading with the words "Rules." These numbers

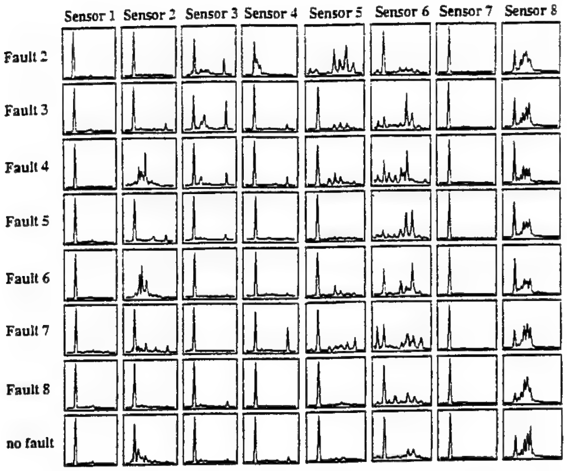


Fig. 7. Power spectrum density plot of 100% torque load with different faults from eight sensors.

Table 3
A list of torque levels, fault types, and number of patterns used in this study

Torque levels (%)	Fault type								Number of patterns
27	N/A[a]	3	4	N/A	N/A	7	8	9	400
40	N/A	3	4	N/A	N/A	7	8	9	700
45	N/A	3	4	N/A	N/A	N/A	8	9	350
50	N/A	3	4	N/A	N/A	7	8	9	700
60	N/A	3	4	N.A	N/A	7	8	9	500
70	N/A	3	4	5	6	7	8	9	500
75	N/A	3	4	5	6	7	8	9	400
80	N/A	3	4	5	6	7	8	9	500
100	2	3	4	5	6	7	8	9	500

[a] N/A, not available.

indicate how many rules or hidden nodes that the neurofuzzy classifier has created. For the last column of Table 4, trained by 100 patterns from every torque levels, the neurofuzzy classifier generated 36 rules (i.e. hidden nodes) with 100% correct classification for all torque levels when testing by all available patterns. It can be found that the proposed neural fuzzy classifier generalizes better at low torque levels.

The proposed neurofuzzy classifier used only one pass to learn all the training patterns. It is worth noting that the hidden nodes, or fuzzy if–then rules, are automatically created equal to the number of fault classes existing for each torque level. This shows that the proposed classifier can deduce the optimum rule bases from this data set. The results have shown that the proposed neurofuzzy classifier performs very well on classifying fault classes with 100% correct classification.

For comparisons, two classifiers were used for comparison with the proposed neurofuzzy classifier for the Westland vibration data set. The first network was the multilayer perceptron (MLP) neural network trained by the backpropagation with variable learning rates. The MLP network was comprised of one hidden layer with 10 hidden nodes and one output layer with four nodes. Log-sigmoidal functions were utilized in the MLP network. The sum of square error (SSE) goal was set to 0.001. The MLP network was trained for 10 trials. To meet the SSE goal. the MLP network used a training time of 414 iterations with 21 min on the average of 10 runs. The second network was the radial basis function (RBF) network constructed using every point of training patterns to set as the centers of Gaussian, i.e. the weight in the radial basis layer. The output weight was determined from linear equation (this was done using function "newrbe" from MATLAB Neural Network Toolbox version 3).

This experiment was performed using 200 patterns of 100% torque level to train the selected classifiers. The testing data sets were composed of the remaining 200 patterns from 100% torque load, 700 patterns from 80% torque, 350 patterns from 75% torque load, and 700 patterns from 70% torque load. The data used were 1128-dimensional vectors that were combined from all eight sensors. To measure the computational complexity, functions "tic" and "toc" in Matlab were utilized to average the learning time with 10 runs on each network. The proposed neurofuzzy network incrementally learned and generated eight clusters i.e., eight rules in the hidden layer. The training time was about 2 min within a single iteration. On this data set, the neurofuzzy spent shorter training time than the MLP and RBF networks did. The computational time was 4 min to construct the RBF network. The results of the simulations are shown in Table 5.

From Table 5, it is found that the neurofuzzy and the RBF network achieved, for 100% torque

Table 4
Percent correct classification of the neurofuzzy classifier for the Westland vibration data[a]

	Training data (torque levels)									
Testing data (Torque levels)	27%	40%	45%	50%	60%	70%	75%	80%	100%	All torque levels
Number of rules	5	5	4	5	5	7	7	7	8	36
27%	100	100	99.5	80	60	79.2	60	59.2	28	100
40%	100	100	100	100	73.2	77	54	40	20	100
45%	86	100	100	100	81.5	62	30.5	25	25	100
50%	87.6	73.6	61.75	100	91	60	40.6	40	40	100
60%	78.6	40	61.75	68.8	100	64	60.2	40	40	100
70%	53.6	40	50	40	60.4	100	99.86	71	44.57	100
75%	40	40	50	40	40	93.14	100	71.43	71.43	100
80%	60	40	50	40	40	45.29	67.71	100	62.14	100
100%	40.4	20	50	40	40	32.86	42.57	62.57	100	100

[a] 100 patterns were used for training when patterns from the same level were used for testing. All patterns from each torque level were used for training when patterns from different levels were used for testing. For the last column, 100 patterns from each torque load level were used for training.

Table 5
Percent correct classification of the MLP, RBFN, and the neurofuzzy classifier, trained by 100% torque level and tested by different torque levels

	Classifier types (trained with 100% torque level)		
	MLP	RBFN	Neurofuzzy
Learning type	Off-line	Off-line	On-line
Learning time	414 epochs, 21 min	1 epoch, 4 min	1 epoch, 2 min
Test level (torque level) (%)			
100	98.75	100	100
80	58.71	16.57	62.14
75	61.14	0.57	71.71
70	37.43	0	46.43

level, 100% correct classification, while the MLP network achieved only 98.75%. For the generalization on different torque levels, i.e. 80, 75, and 70% torque levels, the neurofuzzy shows better performance than the MLP and RBF networks.

5. Conclusions

A novel neurofuzzy network for pattern classification has been proposed. The neurofuzzy classifier based on neural networks and fuzzy set theory is a self-organized classifier. The network automatically deduces fuzzy if–then rules from the numerical data. The proposed classifier synergistically integrates the standard fuzzy inference system and a one-pass supervised learning concept of neural networks. Gaussian radial basis functions are used as the membership functions both in the antecedent and the consequent parts of the fuzzy inference system. The neurofuzzy classifier on-line incrementally learns training patterns with only one pass.

The network has been validated using two benchmark data sets: the Fisher's Iris data set and the Westland vibration data set. The results have shown that on the Iris data set the proposed classifier can classify the testing data set with 97.33% correct classification. For the Westland vibration data, the neurofuzzy classifier achieved 100% correct classification when using the all torque levels for training and testing. It is worth mentioning that the proposed classifier is equipped with an on-line, one-pass, incremental learning rule. As a result, it is suitable for applications that demand to on-line detect new fault scenarios such as in machine condition-based monitoring and any application that prefers a short training time yet classification performance is comparable or even superior to conventional approaches.

To improve the neurofuzzy classifier, some future works are needed to be pursued as follows. First, the proposed classifier is sensitive to the order of the presentation of the training patterns (i.e. different orders of presentation result in different rules and classification performance). It is preferable to use fewer rules with higher classification performance. In order to select a better rule set, a number of trials may be needed by presenting the different random order of training data for each trial. Then, select the rule set that have fewer rules with higher generalization for the test set. This will not spend an expensive computational time, since each trial needs only a quick single iteration for training.

Second, if the network seems to have too many rules, a rule-pruning algorithm should be incorporated into the network to reduce the number of rules. Also the initial standard deviation for the membership function is heuristically chosen, usually set to be a small value such as 0.1. There should be a mechanism that can automatically determine the best choice of the initial standard deviation. Moreover, there is no mathematical proof of convergence of the proposed neurofuzzy network. (Notice that the network is a one-pass and incremental learning algorithm, so it is not easy to derive mathematical proof of convergence.) Finally, using different membership functions both in the antecedents and the consequents, different "AND" and "OR" operators, and different defuzzification methods may improve the classification performance. These tasks are left for future research.

Acknowledgements

This work was supported by the Royal Thai Government and the US Air Force Office of Scientific Research under Grant F49620-98-1-0049.

References

[1] P.K. Simpson, Fuzzy min–max neural networks — part 1: classification, IEEE Transactions on Neural Networks 3 (5) (1992) 776–786.

[2] R.B.W. Heng, M.J.M. Nor, Statistical-analysis of sound and vibration signals for monitoring rolling element bearing condition, Applied Acoustics 53 (1–3) (1998) 211–226.

[3] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, John Wiley and Sons, New York, 1973.

[4] E.A. Ashton, Detection of subpixel anomalies in multispectral infrared imagery using an adaptive Bayesian classifier, IEEE Transactions on Geoscience and Remote Sensing 36 (2) (1998) 506–517.

[5] K. Fukunaga, Introduction to Statistical Pattern Recognition, 2nd Edition, Academic Press, San Diego, 1990.

[6] R.H. Cabell, C.R. Fuller, W.F. Obrien, Neural-network modeling of oscillatory loads and fatigue damage estimation of helicopter components, Journal of Sound and Vibration 209 (2) (1998) 329–342.

[7] W.J. Staszewski, K. Worden, Classification of faults in gearboxes preprocessing algorithms and neural networks, Neural Computing & Applications 5 (3) (1997) 160–183.

[8] L. Zadeh, Fuzzy sets, Information and Control 8 (1965) 338–353.

[9] L.Y.L. Cai, H.K. Kwan, Fuzzy classifications using fuzzy inference networks, IEEE Transactions on Systems Man and Cybernetics Part B — Cybernetics 28 (3) (1998) 334–347.

[10] P. Fu, A.D. Hope, M.A. Javed, Fuzzy classification of milling tool wear, Insight 39 (8) (1997) 553–557.

[11] S.K. Halgamuge, Self-evolving neural networks for rule-based data processing, IEEE Transactions on Signal Processing 45 (11) (1997) 2766–2773.

[12] J.C. Chen, J.T. Black, A fuzzy-nets in-process (fnip) system for tool-breakage monitoring in end-milling operations, International Journal of Machine Tools & Manufacture 37 (6) (1997) 783–800.

[13] V. Chandrasekaran, Z.-Q. Liu, Topology constraint free fuzzy gated neural networks for pattern recognition, IEEE Transactions on Neural Networks 9 (3) (1998) 483–502.

[14] M. Meneganti, F.S. Saviello, R. Tagliaferri, Fuzzy neural networks for classification and detection of anomalies, IEEE Transactions on Neural Networks 9 (5) (1998) 848–861.

[15] S. Li, M.A. Elbestawi, Tool condition monitoring in machining by fuzzy neural networks, Journal of Dynamic Systems, Measurement, and Control — Transactions of the ASME 118 (4) (1996) 665–672.

[16] P. Fu. A.D. Hope. G.A. King. A neurofuzzy classification network and its application. in: Proceedings of the IEEE International Conference on Systems. Man. and Cybernetics, 1998. pp. 4234–4239.

[17] Y.C. Tzeng, K.S. Chen, A fuzzy neural network to SAR image classification, IEEE Transactions on Geoscience and Remote Sensing 36 (1) (1998) 301–307.

[18] E.H. Mamdani. Application of fuzzy algorithms for control of simple dynamic plant. IEEE Proceedings, 121 (12) (1974).

[19] R.A. Fisher. The use of multiple measurements in axonomic problems. Annals of Eugenics 7 (1936) 179–188.

[20] B.G. Cameron. Final Report on CH-46 Aft Transmission Seeded Fault Testing. Westland Research Paper RP907. 1993.

APPENDIX D:

An Effective Neuro-Fuzzy Paradigm for Machinery Condition Health Monitoring

by

Gary G. Yen and Phayung Meesad

IEEE Transactions on Systems, Man and Cybernetics, **31**(4), 2001, pp. 523-536

An Effective Neuro-Fuzzy Paradigm for Machinery Condition Health Monitoring

Gary G. Yen, *Senior Member, IEEE,* and Phayung Meesad, *Student Member, IEEE*

***Abstract*—An innovative neuro-fuzzy network appropriate for fault detection and classification in a machinery condition health monitoring environment is proposed. The network, called an incremental learning fuzzy neural (ILFN) network, uses localized neurons to represent the distributions of the input space and is trained using a one-pass, on-line, and incremental learning algorithm that is fast and can operate in real time. The ILFN network employs a hybrid supervised and unsupervised learning scheme to generate its prototypes. The network is a self-organized structure with the ability to adaptively learn new classes of failure modes and update its parameters continuously while monitoring a system. To demonstrate the feasibility and effectiveness of the proposed network, numerical simulations have been performed using some well-known benchmark data sets, such as the Fisher's Iris data and the Deterding vowel data set. Comparison studies with other well-known classifiers were performed and the ILFN network was found competitive with or even superior to many existing classifiers. The ILFN network was applied on the vibration data known as Westland data set collected from a U.S. Navy CH-46E helicopter test stand, in order to assess its efficiency in machinery condition health monitoring. Using a simple fast Fourier transform (FFT) technique for feature extraction, the ILFN network has shown promising results. With various torque levels for training the network, 100% correct classification was achieved for the same torque levels of the test data.**

***Index Terms*—Fuzzy neural network, incremental learning, machine health monitoring, pattern classification.**

I. Introduction

MODERN engineering technology is leading to increasingly complex machinery with ever more demanding performance criteria. A constant need in prolonging service life and manufacturing yields for global competition calls for an even higher standard in structural reliability. At another extreme, the maintenance and sustainment of aging capital-intensive infrastructures demand innovative technology in condition-based maintenance. A downsized workforce, a declining maintenance budget, and a desire for a "better, cheaper, and smarter" solution have further complicated the risk management decisions. However, currently used diagnostic systems that rely primarily on ingenious sensor innovations or healthy redundant sensor placements to provide early warning are costly, vulnerable and computationally expensive to validate. Specialized nondestructive testing equipment and procedures are often local in nature, heavy, passive, and labor-intensive. State-of-the-art on-board vibration monitoring systems (VMSs) in sophisticated defense vehicles and civic structures serve to collect only vibration spectra or acoustic emissions for off-line analysis. Needless to say, time-critical decisions due to catastrophic failures are often left unresolved. To increase the ability to promptly detect and predict impending failures and catastrophic breakdowns in the complex interrelated structures of plants, vehicles, and processes, a fundamental health usage monitoring (HUM) system using newly emerging computational intelligence technologies was proposed [1]. This system was solely based upon the nondestructive analysis of vibration signatures and acoustic emissions. This model-free integrated approach advanced from frequency analysis and learning theory provided analytical redundancy with respect to the conventional fault detection, identification, and accommodation (FDIA) methodologies that mandate an established high-fidelity dynamic model. The decisions made were then interpreted in order to facilitate expert maintenance procedures for emergency services as well as routine checkups [2].

In the machinery health monitoring system proposed in [1], pattern classification is a key component in identifying failure modes induced from the monitored systems. Service and maintenance can be promptly and correctly performed if the pattern classifier makes an accurate decision. While operating, mechanical components generate some physical parameters, such as temperature or pressure variations, electromagnetic fields, acoustic emissions, or vibration spectra, which contain information about the state of the machinery health. These physical behaviors are sensed by a transducer array to obtain data which is used for failure diagnosis and prognosis. Using pattern classification techniques, signatures (like natural frequency, mode shape, or curvature shape) can be extracted from the data that contains information about machine defects and their causes. With the accurate decision making of a classifier in a monitoring system, machine maintenance can be performed before catastrophic failures occur.

When a machine is operating properly, the physical symptoms, such as vibrations, are generally small and constant. However, when faults develop which lead to significant variations in process dynamics, the physical signatures (e.g., power spectrum density, natural frequency, and mode shape) also vary. To detect these changes, classical off-line iterative learning classifiers are proposed to supervise the monitored system [1], [3], [4]. These classifiers have a drawback in that they generally require a long training time. In addition, gradient-type classifiers often get stuck at local minima and are unable to achieve an optimum solution. Furthermore, while operating, it is possible that

Manuscript received July 23, 1999; revised March 28, 2001. This paper was recommended by Associate Editor P. K. Willett.

The authors are with the Intelligent Systems and Control Laboratory, School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078 USA.

Publisher Item Identifier S 1083-4419(01)05982-9.

novel failure modes are evolving while a monitored system is running. These faults could be significantly different from those known to the classifier. These new classes of defects need to be promptly detected and distinguished from those that have been trained to the classifier. Often, the monitored system generates multiple defects simultaneously. Identification of these multiple defects is also needed in order to perform correct actions for maintenance. Conventional statistical or neural classifiers share known deficiencies in coping with the problems listed above [5]–[8]. In this paper, we propose an effective fuzzy neural network capable of solving the above problems and appropriate for a condition-based health monitoring system. The proposed network advanced from fuzzy ARTMAP architecture [9] incorporates the Gaussian membership functions and provides continuous health monitoring based upon vibration signatures.

For the completeness of the presentation, the remainder of this paper is organized as follows. Section II provides a brief literature survey for pattern classification based on various approaches. The comprehensive understanding of the problem statement and deficiencies of existing literature then leads to the clearly defined objectives of this study given in Section III. Then, the proposed network architecture and the learning rule are introduced in Section IV. To demonstrate the effectiveness and efficiency of the proposed network, numerical simulations and benchmark comparisons are presented in Section V. Finally, Section VI provides some concluding remarks and future research directions.

II. Literature Review

Pattern classification forms a fundamental solution to different problems in real-world applications. The function of pattern classification is to categorize an unknown pattern into a distinct class based upon a suitable similarity measure. Thus, similar patterns are designated to be in the same class while dissimilar patterns are classified into different classes. Engineers and scientists have developed various methodologies to deal with classification problems. These conventional classification techniques make use of statistical decision theory, neural network theory, and fuzzy set theory. Because of their simplicity, statistical approaches, such as Bayesian classifiers [10], are still widely used. To handle more complex classification problems, neural network classifiers, such as the multilayer perceptorn network (MLP) [11], the learning vector quantization (LVQ) [12], and the radial basis function network (RBFN) [13]–[16] with the abilities of parallel computation and nonlinear mapping have been shown to be more suitable because of their learning and generalization abilities. A third classification technique incorporating fuzzy set theory [17] to handle vague information has been extensively applied to pattern classification. The main advantage of fuzzy classification techniques, such as fuzzy-rule-base methods [18], fuzzy c-means [19], fuzzy k-nearest-neighbor [20], [21], and fuzzy decision tree [22], lies in the fact that they provide a soft decision which is a value that describes the degree to which a pattern fits with a class.

The synergetic integration of neural networks and fuzzy sets is also an active area for pattern classification research. A growing number of researchers have designed and exar various forms of fuzzy-neural or neuro-fuzzy networks. idea is to merge the capabilities of model-free and trair systems, parallel computation, and noise tolerance of ne networks with the ability of dealing with imprecise situatior the fuzzy set theory. The integration of neural networks anc fuzzy set theory results in a classifier that has useful prope of both neural networks and fuzzy sets. The combinatio neural networks and fuzzy sets forms a hybrid network handles pattern classification problems very effectively efficiently. Because of their massive parallel computatic units, neural networks have the advantage of fast compu tion so that it is possible to process real-time estimation extensive information. The benefit of fuzzy systems lies their ability to handle the unclear data usually experienced real-world problems [17]. Fuzzy neural networks have sho to be very advantageous in dealing with realistic proble in real-life. Some examples of fuzzy neural networks a neural-fuzzy systems for pattern classification problems a knowledge-based fuzzy MLP [23], neural-network-based fuz classifier [24], neuro-fuzzy system [25], adaptive neural fuz inference system [26], on-line self-constructing neural fuz inference network (SONFIN) [27], fuzzy min–max neur network [28], fuzzy ART neural network [29], fuzzy ARTMA neural network [9], Gaussian ARTMAP neural network [30 and RBF fuzzy ARTMAP neural network [31].

While the other networks cannot learn incrementally, th fuzzy ARTMAP learns new knowledge without having to re learn all the patterns. This concept is appropriate for detectin new faults in machine health monitoring online.

The main concept of the fuzzy ARTMAP is that input pat terns are presented to fuzzy ART$_a$ to be clustered into group while the corresponding targets are presented to fuzzy ART$_b$ also to be clustered into groups. (Fuzzy ART$_a$ and fuzzy ART$_b$ are defined in the fuzzy ARTMAP architecture [9].) Then the two modules are mapped to correct input and output pairs via a map field module. The fuzzy ARTMAP learns to classify inputs by a fuzzy set of features or a pattern of fuzzy membership values between 0 and 1. A hyperbox is used to represent a decision boundary of the input space. Its minimum point and its maximum point define a hyperbox fuzzy set. A membership function is defined with respect to the hyperbox minimum and maximum values in each dimension. Extensive details of the fuzzy ARTMAP neural network are discussed in [9]. Despite the beneficial property of on-line incremental learning, some drawbacks of the fuzzy ARTMAP neural network presented in the literature are as follows:

1) It has no mechanism to avoid overfitting and hence should not be used with noisy data.
2) In the fuzzy ART system, full membership functions are allowed to overlap for each hyperbox, leading to the confusion of decision making (for example a pattern can have full membership in two classes at the same time).
3) It has too many parameters to tune in the network.

Due to above deficiencies, we propose a new network that will adapt to vibration monitoring. The details of specific objectives are given in Section III.

III. Objectives of the Research

The primary objective of this study is to develop a methodology for pattern classification appropriate for machinery condition-based health monitoring applications. This proposed new classifier, advanced from the concepts of the fuzzy ARTMAP concerning on-line incremental learning, is called an *incremental learning fuzzy neural (ILFN)* network. To overcome some of known deficiencies of statistical classifiers, neural classifiers, fuzzy classifiers, and fuzzy-neural network classifiers, the ILFN classifier incorporates the following features.

1) A Hybrid Supervised and Unsupervised Learning Algorithm: A supervised learning algorithm [9], [11]–[14], [32] is used when the corresponding targets are known. On the other hand, when the corresponding target is not available, an unsupervised learning algorithm [12], [16], [19], [20], [28]–[32] is adopted for on-line learning.

2) Fast, On-Line, One-Pass, Incremental Learning: Many well-known neural networks and conventional pattern classification techniques use *off-line* (or batch) learning, which assumes all training patterns and their targets are given. On the other hand, for *on-line* learning only one training pattern is needed at a time. Thus, on-line learning requires less memory than off-line learning does. Off-line learning tends to use longer training time. A *one-pass* learning algorithm has training patterns presented to the classifier only once. *Incremental learning* defines the capability of learning new classes and quickly refining existing classes without forgetting learned information.

3) Ability to Detect New Classes and Label Them Differently from the Existing Corresponding Targets: In some machinery health monitoring systems, such as VMSs, unanticipated fault patterns may develop while the systems are operating. These new patterns need to be promptly detected and learned by the classifiers in order to prescribe a correct action for maintenance. After training, traditional classifiers cannot detect the difference between the learned fault patterns and unseen fault patterns. They often classify the new patterns to the closest learned patterns even when they are significantly different. This may lead to a misunderstanding and cause incorrect service.

4) Ability to Build Decision Boundaries that Separate Nonlinear Separable Problems: Many neural classifiers have overcome the nonlinear separable problems. This new classifier should also provide the ability to build the decision boundaries to separate both linear and nonlinear separable classes.

5) Ability to Make Decision Boundaries of all Overlapping Classes: Bayesian classifiers are generally used to classify overlapping classes; however, constructing the Bayesian classifiers requires knowledge of the probability density function for the classes. Even if the probability density function for each class is unavailable beforehand, the proposed classifiers should be able to classify overlapping classes by employing Gaussian neurons with adaptable variances.

6) Nonparametric Classifier: Parametric classifiers need *a priori* information about the probability density functions of an input pattern space; on the other hand, nonparametric classifiers do not require *a priori* information available [28].

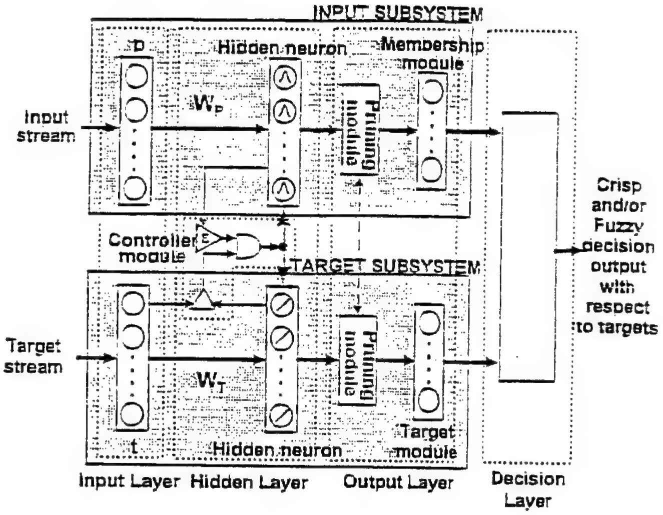


Fig. 1. Network architecture of the ILFN classifier in the supervised learning mode.

7) Ability to Provide Both Soft and Hard Classification Decisions: A hard decision means that a given pattern either belongs to or does not belong to a specific class prototype. In contrast, a *soft* decision allows a given pattern to belong to more than one class prototype with different membership grades [28]. It is possible to detect multiple defects in monitored systems if a soft decision is used.

8) Few Tuning Parameters: Tuning parameters are used for controlling a learning process, and there should be as few parameters as possible to tune in the system [28].

IV. Network Architecture and Classification Algorithm

The ILFN network is advanced from the fuzzy ARTMAP basic idea of on-line and incremental learning behavior. The architecture of the ILFN network is similar to the fuzzy ARTMAP; however, in details, the ILFN network operations are completely different from the fuzzy ARTMAP. While the fuzzy ARTMAP uses hyperbox membership functions, the ILFN network employs Gaussian membership functions that can prevent full membership of overlapping classes. The ILFN network architecture is detailed as follows.

The architecture of the ILFN network is distinguished by two different modes: a supervised learning mode (as shown in Fig. 1) and an unsupervised learning mode (as shown in Fig. 2). The two modes have differences only in the controller module and the target labeling module. Whereas the supervised learning mode requires pairs of input and target of patterns to construct prototypes of the network, the unsupervised learning mode uses the target labeling module to assign the target class for a given input pattern.

The ILFN network has four layers:

1) one input layer;
2) one hidden layer;
3) one output layer;
4) one decision layer, as shown in Figs. 1 and 2.

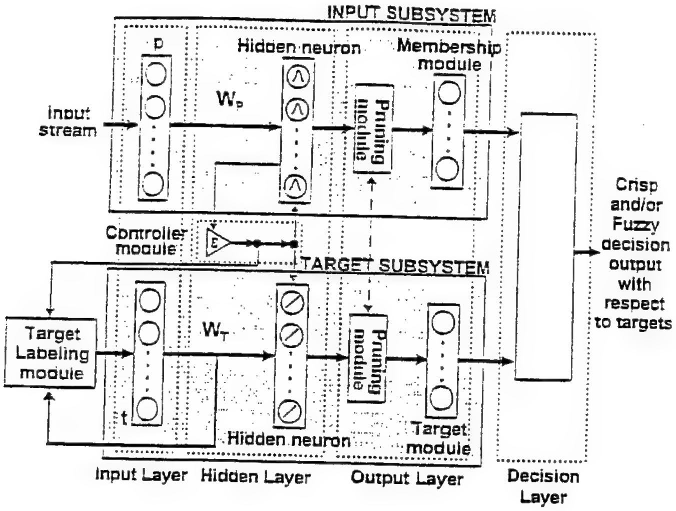


Fig. 2. Network architecture of the ILFN classifier in the unsupervised learning mode.

Generally, the first three layers of the system are composed of two subsystems: an input subsystem and a target subsystem. These subsystems are linked by three connections:

1) the controller module in the hidden layer;
2) the pruning modules in both the input subsystem and the target subsystem of the output layer;
3) decision layer which is the link between the membership module in the input subsystem and the target module in the target subsystem.

The following sections present the details of the input subsystem, the target subsystem, the controller module as well as the fourth layer, the decision layer.

A. Input Subsystem

Fig. 3 illustrates the input subsystem of the ILFN classifier. The input subsystem is composed of three parts:

1) a variable p in the input layer;
2) a Gaussian membership function variable weight W_P in the hidden layer;
3) a pruning module and a membership module in the output layer.

In the input layer, an element of an input vector p connects to each neuron. Neurons of the input layer are fully connected to neurons of the hidden layer via a dynamic synaptic weight matrix W_P, whose rows represent prototype vectors which are the centroids of radial basis functions in the hidden layer. W_P is a trainable weight using learning rules that will be discussed later. W_P grows when a new prototype is detected. Additional rows are added to W_P each time a neuron is added to the hidden layer.

In the hidden layer of the ILFN network, Gaussian membership functions are used. The Gaussian functions are centered on the mean vectors of clusters which are called prototypes of the input pattern space. The Gaussian membership functions are employed to fuzzify the input vectors p, into membership values m_i, with respect to the distance measure between the input vectors p, and the existing prototypes. The membership function at the ith neuron $m_i(\mathrm{p}, \mathrm{w}_{Pi})$, is defined by the following eq tion:

$$m_i(\mathrm{p}, \mathrm{w}_{Pi}) = \exp\left(-\frac{\|\mathrm{p}-\mathrm{w}_{Pi}\|^2}{2\sigma_i^2}\right), \quad i = 1, 2, \ldots, L$$

where $\|\cdot\|$ denotes the Euclidean distance which is used as a sir ilarity measure between two vectors. The weight vector betwe the input layer and the ith hidden neuron w_{Pi} is the center mean vector at the ith neuron in the hidden layer. σ_i represer the standard deviation of the ith neuron in the hidden layer. Ti membership function $m_i(\mathrm{p}, \mathrm{w}_{Pi})$ of the hidden layer is used fuzzify the distance between a given input vector p and the i centers w_{Pi} into a real value m_i which represents the degree (similarity between p and w_{Pi}. The membership functions prc duce localized, bounded, and radially symmetric kernels. Th value of these membership functions monotonically decrease as the distance from the function's center increases.

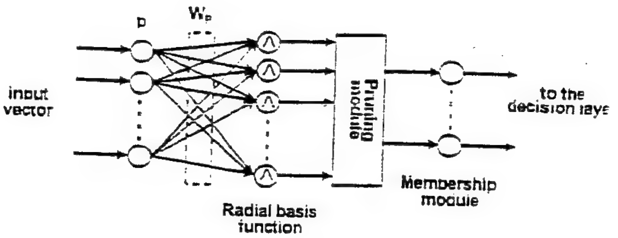


Fig. 3. Input subsystem of the ILFN classifier.

In the ILFN network, a class may have several prototypes These prototypes have different degrees of belonging assigne(to a pattern. Only one prototype with the highest degree of belonging is needed to represent a pattern. The prototypes witl lower degrees of belonging generate redundant classes. To eliminate redundant classes, the pruning module is used in the outpu layer of the ILFN network. Instead of passing many duplicatec classes, only distinguished classes are passed to the membership module. This makes the system easier for human users tc interpret the output.

The pruning procedure of the ILFN network is different from usual pruning procedures that eliminate insignificant neurons or weights [23]. The pruning module used in the ILFN network is a short-term memory. Thus, it performs separately for each input pattern.

In addition, the pruning module in the input subsystem and the pruning module of the target subsystem work in the same way. From each prototype, the maximum membership value in the input subsystem is selected to represent the degree of similarity with respect to a class in the target subsystem.

The membership module in the output layer of the input subsystem receives information transmitted from the pruning module and passes it to the decision layer. The information stored in the membership module is a short-term memory, which means that the information in the membership module differs for different input vectors. Each membership value in the membership module indicates the degree of similarity of an input vector with respect to the target classes of the classifier. The membership values are then mapped in the same order of indexes to classes in the target module in the target subsystem via the decision layer.

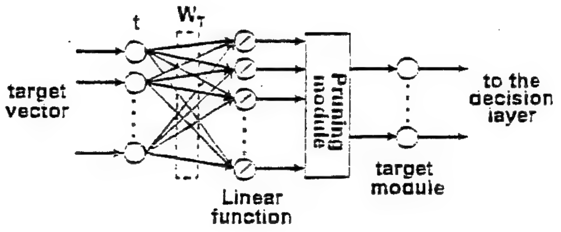


Fig. 4. Target subsystem of the ILFN classifier.

B. Target Subsystem

The target subsystem of the ILFN classifier is depicted in Fig. 4. Each neuron of the input layer in the target subsystem is fully connected to each element of a target vector. A synaptic weight matrix W_T, which needs no training, is used to connect the neurons of the input layer to the neurons of the hidden layer. An additional row is added to W_T when a neuron is added to the hidden layer. These hidden neurons of the target subsystem are activated by linear functions.

As in the input subsystem, the pruning module of the output layer in the target subsystem is used to eliminate redundant classes in the hidden layer. Instead of passing many duplicate classes, only classes with prototypes that have the highest degree of membership for a given input are passed to the membership module. As mentioned before, the pruning module in the target subsystem works the same way as the pruning module in the input subsystem does.

The target module, which is in the output layer of the target subsystem, receives information passed from the pruning module and submits it to the decision layer. Each neuron of the target module is a class or a target of an input vector. The target module is a short-term memory as is the membership module of the input subsystem. In the same order of indexes, the target module is then mapped to the membership module of the input subsystem via the decision layer.

C. Controller Module

The controller module is used to control the growth of neurons in the hidden layers of both the input subsystem and the target subsystem. It operates differently in the controller module in supervised learning mode and unsupervised learning mode.

In the supervised learning mode, there are three components in the controller module: two comparators and one AND gate. One comparator is used to compare the winning membership value from the hidden layer of the input subsystem to the threshold ε. The output of this comparator becomes "true" if the winning membership value is smaller than the ε. This implies that the input vector is significantly different from all existing prototype vectors. The output is sent to one input of the AND gate. Another comparator, which has two inputs, is used to compare the desired target to the predicted output which is stored in the hidden layer of the target subsystem. The output of the comparator becomes "true" if both the desired target and the predicted output are the same. It is sent to another input of the AND gate. If both inputs of the AND gate are "true," its output becomes "true." This allows the system to add one more neuron to the hidden units. In other words, the system generates more neurons whenever the membership value of the winning neuron is smaller than the threshold ε and the desired target and the decision output are the same.

In the unsupervised learning mode, the controller module of the ILFN classifier has only one component which is a comparator. The comparator is used to compare the winning membership value in the hidden layer to the threshold ε. The output of this comparator becomes "true" if the winning membership value is smaller than ε. If the output of the comparator is "true," meaning that a new category is detected, the system adds a new neuron to the hidden layer using the input pattern as the new prototype, then the target labeling module distinguishably assigns a corresponding target to the new prototype.

In addition to a comparator, the controller module in the unsupervised learning mode also has a target labeling module used to assign a target for a new prototype. The target labeling module receives one input from the output of the controller module in the hidden layer of the target subsystem. This input from the controller module tells the target labeling module to assign a target when a new neuron is added to the system. Another input of the target labeling module, representing targets of prototypes, is used to check the existing targets in order to assign a new target that differs from the existing targets.

D. Decision Layer

The decision layer is used to map the membership values in the membership module of the input subsystem to the target classes in the target module of the target subsystem. The output from the decision layer is the output of the system. The decision output can be interpreted as a soft decision or a hard decision. For the soft decision, the decision output assigns different membership values to the pattern classes or prototypes. This allows a given pattern to belong to more than one class with different degrees of similarity measure. For the hard decision, the decision output selects only one class with the highest membership value.

E. ILFN System Dynamics

Both W_P and W_T are allowed to grow when the system detects new classes. However, only W_P can adaptively change its information or learn new prototypes. At the initialized state, there are no neurons in the hidden layer. The first neuron in the hidden layer is set up after the first input vector p is presented to the input subsystem of the network while the first target vector t is presented to the input layer in the target subsystem. Then, both W_P and W_T set up the first neuron using p and t, respectively. The next input vector is compared to the existing prototype. If there is a significant difference (depending on the threshold ε), then a new neuron is added to the hidden layer; p is added to W_P, and t is added to W_T. On the other hand, if the input vector meets the similarity criterion then, instead of adding a new neuron, the learning process is performed. The W_P and other parameters are updated to include the new data in the existing prototypes.

F. Learning Process

The learning process takes place only in the hidden layer. It adapts the synaptic weight W_P and updates the variables regarding the pattern clusters of the input space. In the learning process, each input vector p from the input space is fuzzified to a membership value at each node of the hidden layer with respect to the distance measure between input vector p and the synaptic weight matrix $\mathbf{W_P}$. The winning node of the hidden layer is determined by the defuzzification process using the fuzzy OR operation ($\vee$) defined as

$$\text{winner} \equiv m_1 \vee m_2 \vee \cdots \vee m_L \tag{2}$$

$$J \equiv \text{winner index} = \arg \max_i (m_i) \tag{3}$$

where $m_1 \vee m_2 = m_1$ if $m_1 > m_2$; $m_1 \vee m_2 = m_2$ if $m_1 < m_2$. Only the parameters of the winner node (i.e., Jth neuron) including count, mean, and standard deviation are updated, while other losing nodes remain the same, as follows:

$$C_{J,\text{new}} = C_{J,\text{old}} + 1 \tag{4}$$

$$\mathbf{w}_{PJ,\text{new}} = \frac{\mathbf{w}_{PJ,\text{old}}(C_{J,\text{new}} - 1) + \mathbf{p}}{C_{J,\text{new}}} \tag{5}$$

$$\sigma_{J,\text{new}} = \begin{cases} \sqrt{\left(1 - \frac{1}{C_{J,\text{new}}}\right)\sigma^2_{J,\text{new}} + \frac{(\sigma_{J,\text{new}} - \mathbf{p})^2}{C_{J,\text{new}}}} \\ \text{if } C_{J,\text{new}} > 1, \\ \sigma_0 \\ \text{otherwise} \end{cases} \tag{6}$$

where a parameter with the subscript "old" represents that parameter before updating and a parameter with the subscript "new" represents that parameter after updating. C_J represents the number of patterns that have been counted into the Jth prototype. The mean $\mathbf{w}_{PJ}$, the center, or the Jth prototype, is a row in the synaptic weight $\mathbf{W_P}$. The standard deviation σ_J will be used to indicate the spread of the data in the Jth prototype. σ_0 is the initial standard deviation representing the isotropic spread in pattern space of a new category for the first sample. σ_0 is usually chosen small enough (e.g., a value between 0.001 and 0.05) to include only the pattern that is setup for the new prototype. After the patterns near the prototype are included in the same prototype, the standard deviation σ_J is updated accordingly.

Equations (4)–(6) are learning rules used to update the prototype variables in the input subsystem. The number of patterns belonging to each cluster is updated by (4). By knowing the previous centers and the number of patterns that belong to a cluster, new centers can be calculated by (5). The estimated standard deviations [30] can be calculated if the previous standard deviation and the number of the patterns belonging to a cluster are known. Estimated standard deviations, which are the spread of the Gaussian membership functions, are determined by (6).

G. Decision Boundaries

The purpose of pattern classification is to determine to what class a given sample belongs. Through an observation or measurement process, a set of numbers which make up the observation vector is obtained. The observation vector serves as the input to a decision rule by which the sample to one of the given classes is assigned.

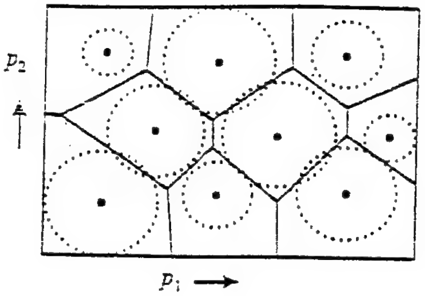


Fig. 5. Decision boundaries among prototypes of the ILFN classifier.

The decision boundaries of the ILFN network distinguish among prototypes in the Voronoi tessellation [12]. Each prototype has its own region separated by the decision boundaries. Since the ILFN classifier uses Gaussian-type membership functions with different standard deviations, the soft decision boundaries of the ILFN classifier are quadratic. However, the hard decision boundary between the neighboring prototype vectors is a hyperplane containing the points that have the same degree of the membership value, as shown in Fig. 5. Fig. 5 shows the decision boundaries among prototypes of the ILFN network in which dotted circles indicate the spread of statistical data for each prototype.

H. Classification Algorithm

The ILFN network can learn in two different ways: 1) supervised learning, which requires both input patterns and corresponding targets and 2) unsupervised learning, which requires only input patterns without corresponding targets and in which the target labeling module will assign appropriate class labels. The classification algorithm of the ILFN classifier is listed as follows:

Step 1) Set the user-defined threshold parameter (ε), the initial standard deviation σ_0, and the maximum number of patterns allowed in each cluster.

Step 2) Retrieve the first input pattern

- Use the first input pattern p to set up the first prototype (or mean) to $\mathbf{W_P}$.
- Set the number of patterns for the first node to be 1.
- Set the standard deviation equal to the initial standard deviation, σ_0.
- Set a new neuron to $\mathbf{W_T}$ using the first target t to be the corresponding target of the prototype in $\mathbf{W_P}$.

Step 3) Retrieve the next training sample with an input and target.

Step 4) Measure the Euclidean distance between the input p and the prototype $\mathbf{W_P}$.

Step 5) Calculate membership values for each node using the Gaussian-type radial basis function.

Step 6) Assign membership values to each node. The current input pattern has different degrees for each node or prototype. For each class, select the maximum membership value from each prototype to represent the degree of similarity with respect to that class.

Step 7) Identify the largest membership using the fuzzy OR operator.

Step 8) For the winning node:

If there is the corresponding target (i.e., supervised learning mode):

a) If the winner is larger than ε and the target t is the same value as $\mathbf{W}_T$ at the winner node then update weight $\mathbf{W}_P$, the standard deviation, and the number of patterns belonging to this node.

b) If 1) is not satisfied, then:

- Set a new node center for $\mathbf{W}_P$ using the input pattern p.
- Set the number of patterns for the new node to be 1.
- Set the initial standard deviation to the new node.
- Add a new neuron to $\mathbf{W}_T$ using the new target t as the corresponding target of a new prototype in $\mathbf{W}_P$.

If there is no corresponding target (i.e., unsupervised learning mode):

a) If the winner is larger than ε and the number of patterns is less than the maximum number of allowed patterns, then update the weight $\mathbf{W}_P$, the standard deviation, and the number of patterns belonging to this node. Identify the class output which is stored in $\mathbf{W}_T$ at the same index of the winner node of $\mathbf{W}_P$.

b) If the winner is smaller than ε, then:

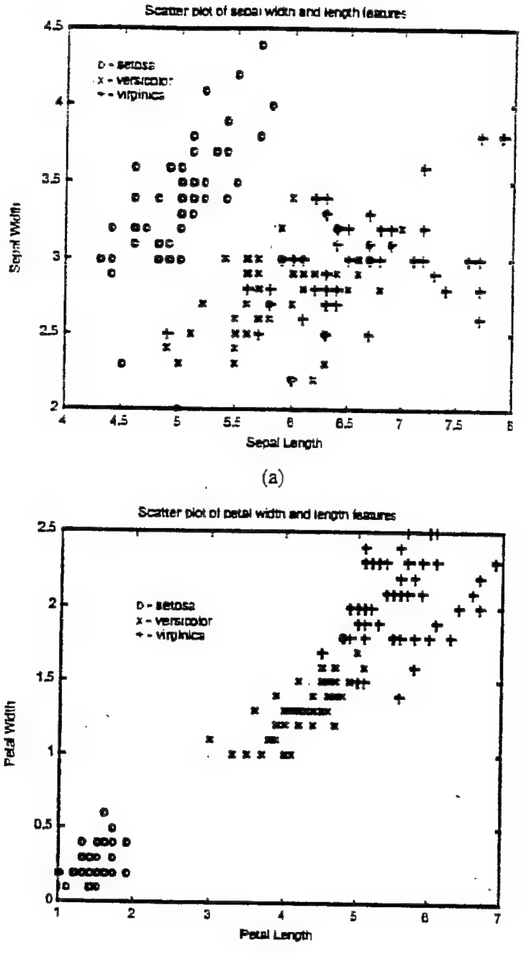
- Set a new node center for $\mathbf{W}_P$ using the input pattern p.
- Set the number of patterns for the new node to be 1.
- Set the initial standard deviation to the new node.
- Add a new neuron to $\mathbf{W}_T$ and assign a new target as the corresponding target of a new prototype in $\mathbf{W}_P$. (The assigned new target must be significantly different from the existing targets already stored in $\mathbf{W}_T$. For example, if there exist targets in $\mathbf{W}_T = [1\,2\,3]^T$, the new target should be "4," that is $\mathbf{W}_T$ becomes $[1\,2\,3\,4]^T$.)

Step 9) If there are no more input patterns, then stop. Otherwise, go to Step 3.

Usually, if the user knows both input patterns and their targets, the network is trained in the supervised learning mode. After supervised training, the network is used in a pattern classification system. The ILFN network can detect new categories that have not been trained. When the system detects new categories, it employs the unsupervised learning mode by using the target labeling module to assign the corresponding targets to the input patterns. The targets that are assigned to the novel prototypes are significantly different from the existing targets in the target module.

V. Simulations and Results

To demonstrate the performance of the ILFN classifier, numerical simulations were used in our experiments. The simulation programs were written to run under MATLAB on a Pentium 233MMX PC. Three data sets were used for training and testing the classifier in our studies. The first benchmark data set was the well-known Fisher's Iris data set [33]. The second data set was a vowel data set [34]. The vowel data set is electronically available from the connectionist benchmark collection at Carnegie Mellon University, Pittsburgh, PA [35]. For the first two data sets used in this study, the results have shown that the ILFN classifier is capable of learning on-line real-time in only one pass through all training data. In addition, the prediction capability of the ILFN classifier was found to be as good as or even better in many cases than many existing classifiers. With the ability of fast, one-pass, on-line, real-time, incremental learning, the ILFN network is found to be applicable in real-world applications. The last and most important data set was a time-series vibration data set known as Westland vibration data [36]. The details of the three experiments are as follows.



(a)

(b)

Fig. 6. (a) Scatter plot of sepal width and length features of the Fisher's iris data. (b) Scatter plot of petal width and length features of the Fisher's iris data.

A. Fisher's Iris Flower Data Set

The Fisher's Iris flower data set consists of 150 patterns and four features:

1) sepal length;
2) sepal width;
3) petal length;
4) petal width.

The four features describe the shape and size of the irises. Each pattern in the data set falls into one of three classes:

1) Setosa;
2) Versicolor;
3) Virginica;

TABLE I
COMPARISON PERFORMANCE BETWEEN THE ILFN CLASSIFIER AND FUZZY ARTMAP

Trial number	ILFN Classifier				Fuzzy ARTMAP			
	Number iterations	Hidden nodes	% correct training set	% correct test set	Number iterations	Hidden nodes	% correct training set	% correct test set
1	1	6	94.67	94.67	1	2	100	92
2	1	7	97.33	98.67	3	6	100	90.67
3	1	4	97.33	96	3	6	100	93.33
4	1	5	96	97.33	2	5	100	96
5	1	6	97.33	93.33	2	6	100	93.33
6	1	6	97.33	98.67	4	6	100	94.67
7	1	6	94.67	94.67	1	4	100	92
8	1	7	97.33	98.67	3	6	100	90.67
9	1	8	98.67	97.33	2	5	100	93.33
10	1	6	96	93.33	2	5	100	94.67
11	1	5	96	96	2	5	100	96
12	1	6	97.33	93.33	2	6	100	93.33
13	1	7	98.67	98.67	2	5	100	96
14	1	6	94.67	94.67	1	4	100	92
15	1	6	96	98.67	2	4	100	94.67
16	1	7	97.33	98.67	3	6	100	90.67
17	1	6	96	94.67	1	4	100	92
18	1	5	96	96	2	5	100	94.67
19	1	6	97.33	94.67	2	6	100	93.33
20	1	7	98.67	97.33	2	5	100	96
Average	1	6.1	96.733	96.268	2.1	5.15	100	93.467

Remark: Fuzzy ARTMAP used in this study is in the Art Gallery version 1, written by Lar Liden. The Art Gallery was obtained from ftp://cns-ftp.bu.edu/pub/ART_GALLERY/Windows/win_gal.zip.

with a total of 50 patterns per class. For the purpose of this experiment, we will call them Class 1, Class 2, and Class 3, respectively. Class 1 is linearly separable from the other two. However, Class 2 and Class 3 are not linearly separable from each other.

Fig. 6(a) shows the scatter plot of Iris data for sepal width and length features. It is worth noting from the plot that Class 1 can be easily separated from Class 2 and Class 3. However, Classes 2 and 3 seem very difficult to separate since there is an overlap between them. Moreover, in Fig. 6(b), the petal width and length features are plotted showing that Class 1 is very well separated from Class 2 and Class 3. Again, Class 2 and Class 3 remain overlapped [33].

1) A Comparison Between the ILFN Classifier and the Fuzzy ARTMAP: In this study, the training data set was composed of the first 25 patterns of each class, while the testing data set was composed of the remaining 25 patterns of each class. Twenty trials were performed. For each trial, the presentation order of the training data was randomly selected. To compare the performance of the ILFN network with a similarly supervised on-line incremental learning classifier, the fuzzy ARTMAP neural network was used. The ILFN classifier and the fuzzy ARTMAP were trained with the same training data set. Then, both networks were tested for generalization using the same testing data. For the parameters of the ILFN classifier, the initial standard deviation σ_0 was set to 0.001 and the threshold ε was set between 0 and 1. When given the same order of presentation of this data set, the ILFN network yields the same number of hidden neurons and the same classification performance independent of ε chosen. The parameters of the fuzzy ARTMAP were set as follows: the vigilance parameters $\rho_a = 0.5$ and $\rho_b = 0.5$, and the learning rate $\beta = 1$. The results of the study are shown in Table I.

From Table I, using the testing data, the IFLN achieved a maximum of 98.67%, an average of 96.268%, and a minimum of 93.33% correct classification. On the other hand, the fuzzy ARTMAP classifier achieved a maximum of 96%, an average of 93.467%, and a minimum of 90.67% correct classification. Moreover, the ILFN classifier used only one-iteration learning through all training data while the fuzzy ARTMAP used one to four iterations to learn the training patterns. However, both algorithms used training times within only a few seconds. For this data set, the number of nodes of the ILFN classifier was not sensitive to the threshold value ε; thus, different values of ε (between 0 and 1) yielded the same number of hidden neurons and the same performance of correct classification. For the fuzzy ARTMAP, the number of hidden neurons was very sensitive to the choices of vigilance parameters; ρ_a and ρ_b.

2) Comparisons Among Other Classifiers: Table II shows the classification performance among other classifiers using the Fisher Iris data. The classifiers in row one to row six, reported by Simpson [28], show that most of the classifiers were able to predict testing data with the number of incorrect classification from two to four. (See details in [28] on how to construct the training and the testing data for these experiments.) It is worth mentioning that those classifiers, except the fuzzy min–max classifier, cannot learn on-line. The fuzzy min–max classifier, which is an unsupervised algorithm, uses hyperboxes for representing the input distribution. On the other hand, the ILFN classifier uses the localized Gaussian function which is more appropriate to represent the distribution of data space [30]. The summary results of the ILFN network and the fuzzy ARTMAP in this study are also included in the last two rows of Table II.

B. Vowel Recognition Data Set

For the Deterding vowel recognition data [34], four male and four female speakers were used for training, and an additional four male and three female speakers were used for testing. The data set is in ten-dimensional input space with 528 samples for the training set and 462 samples for the testing set.

TABLE II
COMPARISON OF PERFORMANCE AMONG EXISTING CLASSIFIERS FOR IRIS DATA

Technique	No. Wrong	Remarks
Bayes classifier*	2	Very amenable to this type of data.
k-nearest neighbor*	4	Scales up poorly.
Fuzzy k-NN*	4	Allows fuzzy labels for data points.
Perceptron*	3	Limited to linear discrimination.
Fuzzy perceptron*	2	Fuzzifies linear boundaries.
Fuzzy min-max *	2	Single pass learning, learns on-line. (hyperbox distribution).
Fuzzy ARTMAP	2-6 (Run 20 trials)	Learns on-line with 1 to 4 passes less than one second. Uses hyperbox distribution.
ILFN classifier	1-5 (Run 20 trials)	One-pass on-line learning in less than one second. Uses Guassian distribution.

* According to Simpson in [28]

TABLE III
ILFN CLASSIFIER PERFORMANCE ON VOWEL DATA USING DIFFERENT VALUES OF THE THRESHOLD

Threshold	# Hidden nodes	Training time	% Correct of training data	% Correct of test data
10^{-1}	127	2.48 S	99.05	52.81
10^{-4}	101	1.92 S	99.05	54.98
10^{-8}	**90**	**1.92 S**	**98.67**	**57.36**
10^{-12}	82	1.92 S	98.48	54.55
10^{-20}	78	1.92 S	97.54	53.9

TABLE IV
VOWEL CLASSIFICATION WITH DIFFERENT NONLINEAR CLASSIFIERS [37]

Classifier	# hidden units	# correct	percent correct
Single-layer	N/A	154	33
Multi-layer	88	234	51
Multi-layer	22	206	45
Multi-layer	11	203	44
Modified Kanerva	528	231	50
Modified Kanerva	88	197	43
Radial basis function	528	247	53
Radial basis function	88	220	48
Gaussian node network	528	252	55
Gaussian node network	88	247	53
Gaussian node network	22	250	54
Gaussian node network	11	211	47
Square node network	88	253	55
Square node network	22	236	51
Square node network	11	217	50
Nearest neighbor	N/A	260	56
ILFN	90	265	57.36

Due to the nature of the vowel data set, which is extremely difficult to separate, a wide range of threshold values was used. In our study, which uses different thresholds ranging from 10^{-1} down to 10^{-20}, the ILFN classifier generated hidden neurons ranging in number from 127 down to 78, as shown in Table III. Larger thresholds allowed the classifier to create more neurons than smaller thresholds. However, the larger number of neurons in the hidden layer does not imply a better performance in predicting the testing data.

Table III shows the ILFN classifier performance on vowel data using different values of the threshold ε. The classifier performed on the test data in various percentages of correct prediction. When using the threshold of 10^{-1}, 127 hidden nodes were generated and the correct prediction of testing data was only 52.81%. When using the threshold of 10^{-4}, only 101 hidden nodes were generated and the correct prediction of testing data was improved to 54.98%. Using the threshold of 10^{-8}, the ILFN classifier was able to classify with the best generalization of 57.36% for the number of hidden nodes of 90. Again, when thresholds smaller than 10^{-8} were used, the percent of correct prediction decreased gracefully. The proposed IFLN classifier was trained in one pass through all data with an average training time of less than 2 s.

The vowel classification using various nonlinear classifiers is shown in Table IV. The comparison study was performed by Robinson [37]. In Robinson's study, the best results with the correct prediction of 56% were obtained using the nearest neighbor classifier. The IFLN can achieve 57.36%.

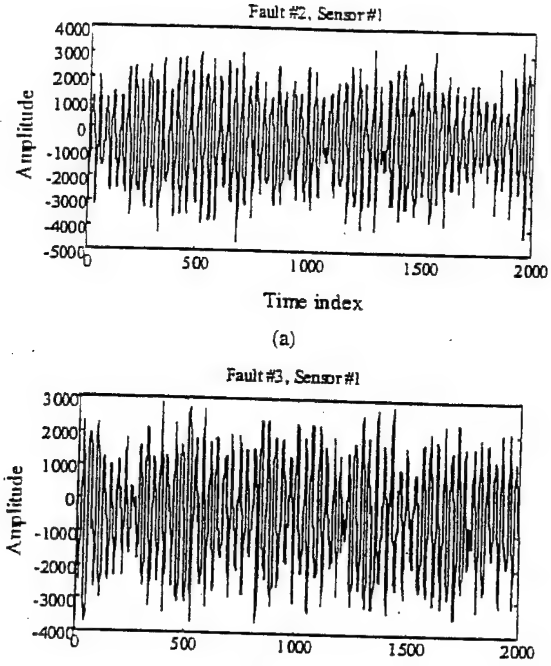
C. *Westland Vibration Data Set*

This data set consists of vibration data recorded using eight accelerometers mounted on different locations on the aft main power transmission of a U.S. Navy CH-46E helicopter. The CH-46E Chinook is a twin-rotor, fore/aft transmission rotorcraft powered by two turbine engines. The data set was archived at the Applied Research Laboratory (ARL) of Penn State University. The vibration data set was collected by using an International Recording Instruments Group analog tape recorder and a single mixbox and aft main transmission installed on a test stand and run at nine different torque levels (i.e., 100%, 80%, 75%, 70%, 60%, 50%, 45%, 40%, and 27%). While collecting the data, only one faulted component was installed in the mixbox and transmission. Then, vibration data was recorded for many types of faults listed in Table V. Employing a ten-channel data acquisition system, the data was then digitized at a sample rate of 103 116.08 Hz with a 16-bit quantization level and saved in 1.506-MB data files. All together, there are 71 files; each file contains all eight accelerometer signals. The data files used in this study were 1 s data files [36].

1) Westland Data Characteristics: Fig. 7(a) and (b) shows two samples of vibration data in the time domain pertaining to Fault Class 2 and Class 3 from Accelerometer 1 of the Westland Data Archive. As clearly seen, it is difficult to discriminate the two raw time-series data. Since the raw time series data provide little information to use for classification, it is preferable to transform the signal from time domain to frequency domain. The vibration signatures in frequency domain are shown in Fig. 8(a) and (b), which are power spectral density plots of the two signals given in Fig. 7(a) and (b), respectively. It is easy to see that frequency contents above 20 kHz are less useful. The effective information for classification is in the frequency range of 3 kHz–10 kHz. For the frequency range of 0–12 kHz, Fig. 9(a) and (b) provide a "zoom-in" version of the power spectrum density plot shown in Fig. 8(a) and (b), respectively. More sample plots in the frequency domain of 100% torque level of the Westland vibration data are shown in Fig. 10. Fig. 10 shows sample patterns of Fault 2, Fault 3, Fault 4, Fault 5, Fault 6, Fault 7, Fault 8, and No Fault from all eight accelerometers. It is worth

TABLE V
LIST OF THE FAULT TYPES CREATED IN THE TEST GEARBOX

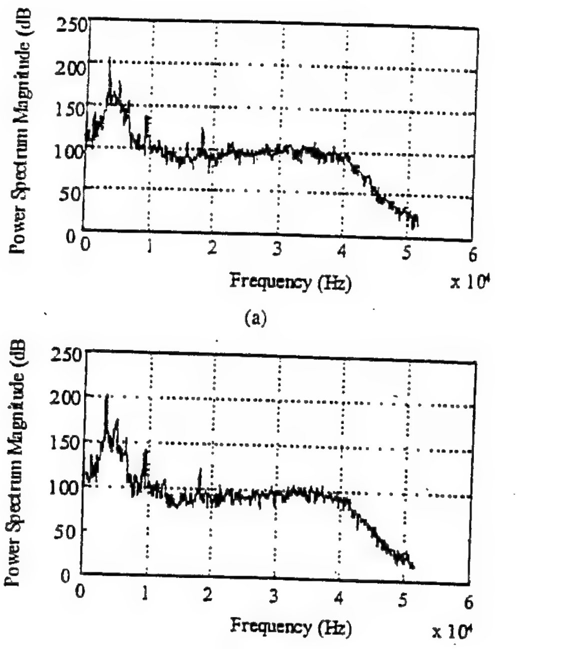
Fault #	Description
2	Epicyclic Planet Gear Bore/Bearing/Inner Race Corrosion Spalling
3	Spiral Bevel Input Pinion Bearing Journal Corrosion Pitting/Spalling
4	Spiral Bevel Input Pinion Gear Tooth Spalling/Scuffing
5	High Speed Helical Input Pinion Tooth Chipping and Freewheel Clutch Bearing False Brinnelling
6	Helical Idler Gear Crack Propagation
7	Collector Gear Crack Propagation
8	Quill Shaft Crack Propagation
9	No Defect



(a)

(b)

Fig. 7. (a) Plot of time series data of Fault 2 from Sensor 1. (b) Plot of time series data of Fault 3 from Sensor 1.



(a)

(b)

Fig. 8. (a) Power spectrum density dB plot of Fault 2 from Sensor 1. (b) Power spectrum density dB plot of Fault 3 from Sensor 1.

noting that data from each sensor alone is not sufficient to classify the fault classes. Moreover, it is easier to classify the data by using all patterns obtained from the eight sensors. Integrating fault patterns from all eight accelerometers is more informative for classification. In this study, most of our experiments used the combined signatures from all eight sensors as training patterns.

2) Using ILFN Classifier on Westland Vibration Data: In our experiments, vibration time-series data was preprocessed using the fast Fourier transform (FFT) technique to transform from the time domain to the frequency domain. A Hanning window of 1024 samples was utilized. We filtered the data with the interested frequency band of 3 kHz–10 kHz, getting a 141×1 vector for each channel. Vectors from the eight channels were set into one vector (1128×1 vector). Then, they were used as training data for the ILFN classifier as well as the other classifiers used in this study. The Fault types and torque levels of the Westland vibration data used in the experiments are shown in Tables VI and VII. The summary results from the experiments are given in Table VIII.

Table VIII shows the results from all simulations in our studies. In the simulations of the Westland data set, all torque load levels (i.e., 100%, 80%, 75%, 70%, 60%, 50%, 45%, 40%, and 27%) were used to train the ILFN classifier. Only ten patterns were used for training, and the remaining patterns were used for testing in each torque level. All patterns were used for training when different torque load levels were used for testing. For the last column of Table VIII, the training set was composed from the first ten patterns of each torque level. The threshold value was selected between 0 and 0.7 and the initial standard deviation selected was 0.001. This yielded the same number of hidden neurons and the same classification performance.

In Table VIII, the columns represent the "training data" with different torque levels, and the rows indicate the "testing data" with different torque levels. The percent of correct classification results are interpreted by crossing each column with each row. For instance, 100% correct classification was achieved when the ILFN network was trained by the 40% torque level and was tested by the 50% torque level. The numbers of hidden neurons resulting from the training process of the ILFN classifier are

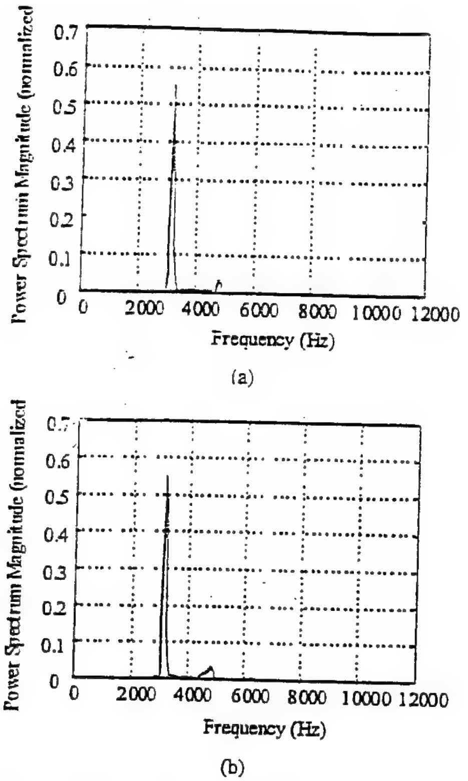


Fig. 9. (a) Power spectrum density plot of Fault 2 from Sensor 1. (b) Power spectrum density plot of Fault 3 from Sensor 1.

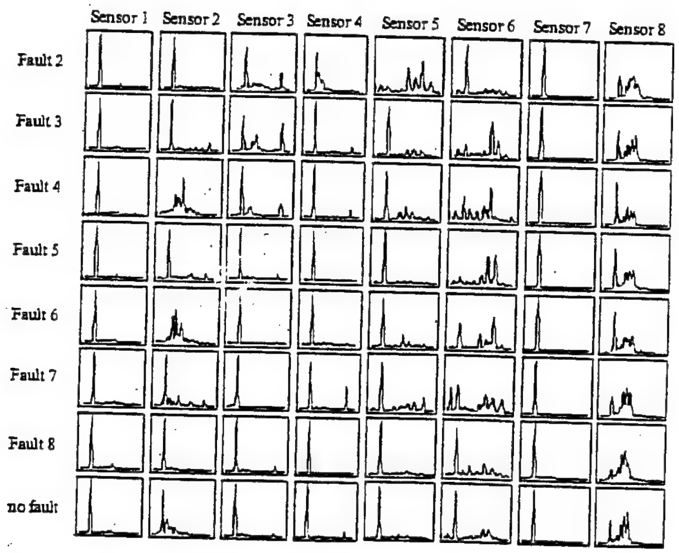


Fig. 10. Power spectrum density plot of 100% torque load with different faults from eight sensors.

shown in Table VIII in the row "Hidden nodes." These numbers indicate how many prototypes the ILFN classifier has created.

Using ten patterns of the 100% torque level for training, the classifier created eight neurons in the hidden layer. The ILFN classifier obtained 100% correct classification using the remaining data of the same torque load for testing. Moreover, using all 400 patterns of the 100% torque level for training, the ILFN network also created eight neurons in the hidden layer. The other torque levels were used to test the performance of the ILFN network. The correct classification of 71.43% was achieved for the 80% torque load; for the 75% torque load, 81.42% correct prediction was obtained. The ILFN classifier yielded the correct classification of 57.14%, 41.2%, 37.2%, 42.5%, 4%, and 7.4% for torque levels of 70%, 60%, 50%, 45%, 40%, and 27%, respectively.

It is worth noticing that when the same torque level was used both for training and testing, the ILFN classifier achieved 100% correct classification. (Note that testing patterns were different from the training patterns, i.e., obtained from different time series, but they were in the same torque level.) Furthermore, using high torque levels (i.e., 100% and 80%) for training, in the testing phase, the ILFN classifier achieved perfect classification only when the testing patterns from the same torque level were used. However, using 75%, 70%, 60%, 50%, 45%, 40%, or 27% torque level for testing, the ILFN network was able to correctly classify a larger range of torque levels. For example, when the ILFN classifier was trained by using a 50% torque level, 100% correct classification was obtained from the range of 40% through 50% torque levels.

3) Comparisons Among Other Classifiers: More experimental results on Westland vibration data are shown in Table IX, which shows the comparison among the MLP, RBFN, LVQ, and ILFN classifier. This simulation was performed using 200 patterns of 100% torque levels to train each classifier. The testing data sets were composed of the remaining 200 patterns from 100% torque load, 700 patterns from 80% torque, 350 patterns from 75% torque, and 700 patterns from 70% torque load. The data used were 1128-dimensional vectors that were combined from all eight sensors. CPU usage time averaged over 50 runs. The results are shown in Table IX.

The first network was the MLP trained by the Backpropagation with variable learning rates. The MLP was comprised of one hidden layer with ten hidden nodes and one output layer with four nodes. (Output targets are labeled in binary form.) Logsigmoidal functions were utilized in the MLP network. The sum of square error (SSE) goal was set to 0.001. The MLP was trained for 50 trials. To meet the SSE goal, the MLP network used a training time of 475 iterations with 400 s on the average of 50 runs. We noticed that for many trials, the MLP was stuck at some local minima, unable to converge to the specified goal. The second network was the RBF network. Using one-pass self-selection of the hidden centers by a successive approximation method [38], the RBFN constructed eight hidden neurons in the hidden layer. Then, the output weight was determined using the method proposed by Haykin [14]. The RBFN quickly learned within a single iteration. The third network was the LVQ network. The network was composed of an eight-neuron LVQ layer and a four-neuron linear layer. The maximum training time of the LVQ was set to be 500 iterations. The LVQ used approximately 194 s for training on the average of 50 runs.

The ILFN classifier incrementally learned and generated eight neurons in the hidden layer. The training time was about 4 s within a single iteration. On this data set the ILFN network used a training time approximately 100 times, three times, and 64 times faster than the MLP, the RBFN, and the LVQ, respectively. For the generalization capacity, it was shown that the ILFN classifier was competitive with the LVQ. In Table IX, both the ILFN classifier and the LVQ were able to classify the

TABLE VI
FAULT TYPES AND TORQUE LEVELS OF WESTLAND VIBRATION DATA USED IN THE EXPERIMENTS

Fault types	Torque levels								
	100%	80%	75%	70%	60%	50%	45%	40%	27%
2	USED	NA	NA	NA	NA	NA	NA	NA	NA
3	USED	USED	USED	USED	USED	USED	USED	USED	USED
4	USED	USED	USED	USED	USED	USED	USED	USED	USED
5	USED	USED	USED	USED	NA	NA	NA	NA	NA
6	USED	USED	USED	USED	NA	NA	NA	NA	NA
7	USED	USED	USED	USED	USED	USED	NA	USED	USED
8	USED	USED	USED	USED	USED	USED	USED	USED	USED
9 (no fault)	USED	USED	USED	USED	USED	USED	USED	USED	USED

NA=Not available

TABLE VII
TORQUE LEVELS AND THE NUMBER OF PATTERNS USED IN THE EXPERIMENTS

Torque levels	100%	80%	75%	70%	60%	50%	45%	40%	27%
# patterns	400	700	350	700	500	500	400	500	500

100%-torque-load testing data with 100% correct classification. The percent correct classification of the two networks was reduced to 71%, 74%, and 57% when using 80%, 75%, and 70% torque load for testing, respectively. However, considering the capability of on-line, real-time, incremental learning, the ILFN network was superior to the LVQ. Moreover, based on generalization and fast on-line learning ability, the ILFN classifier was superior to the MLP and to the RBFN off-line learning algorithms. This result emphasizes that the ILFN classifier is suitable for machinery condition health monitoring systems which require a fast on-line real-time learning algorithm.

4) ILFN Learning Simulation in an Unsupervised Learning Mode: To study the ability of the ILFN classifier in an unsupervised learning mode, which is usually used when the ILFN network is monitoring a system, we used 100% torque load to train the ILFN network. First, only a "No Fault" class was trained to the network. Acting as a monitoring system, the ILFN classifier repeatedly received unseen patterns in order to classify them. The ILFN network was able to detect new classes. It learned the incoming faults by creating new neurons and designating new targets for the unseen patterns that were significantly different from the patterns that had been learned.

Table X illustrates the performance of the ILFN network in an unsupervised learning mode. In Table X, first the ILFN classifier was trained using class "No Fault" with the corresponding target "0000." Then, patterns from Fault 8, Fault 5, Fault 2, Fault 3, Fault 6, Fault 7, and Fault 4 were presented to the ILFN network without corresponding targets. The ILFN network assigned targets to be "0001," "0010," "0011," "0100," "0101," "0110," and "0111," respectively. In order to have different targets with the existing targets, first the ILFN classifier checked the existing targets finding the highest number in the target module. Then, using the increment of the highest number by one, the ILFN classifier assigned the new target to the incoming pattern.

VI. CONCLUSIONS

A new algorithm based on fuzzy neural networks called *incremental learning fuzzy neural (ILFN)* network has been developed for pattern classification. The ILFN classifier employs a hybrid supervised and unsupervised learning scheme to generate its prototypes. The network is a self-organized classifier with the capability of adaptively learning new information without forgetting existing information. The classifier can detect new classes and update its parameters while monitoring a system. Moreover, it utilizes fast real-time on-line learning without knowing *a priori* information. In addition, it has the capability to make both soft (fuzzy) and hard (crisp) decisions and is able to classify both linear separable and nonlinear separable problems.

The network is a synergetic integration of fuzzy sets and neural networks. It employs the fast parallel computation and learning capability of neural networks. In addition, fuzzy set theory adds the ability to represent and manipulate imprecise information.

Three benchmark data sets (the Fisher's Iris data set, the Deterding vowel data set, and the Westland vibration data set) were used in simulations to demonstrate the performance of the ILFN classifier. Comparisons between the ILFN classifier and some existing methods were made. The results show that, in terms of classification performance, the ILFN classifier is competitive with or even better than many well-known classifiers, including the MLP, the RBFN, the LVQ, and the Fuzzy ARTMAP classifier. Additionally, in terms of training time, the ILFN network is superior to classical classifiers. Furthermore, the on-line, real-time, one-pass, incremental learning behavior allows ILFN network to detect new classes and update its parameters without using old data to retrain the network. The ILFN classifier, acting as a component in a monitoring system, was used extensively to investigate the Westland vibration data. The results from the simulation studies have shown that the real-time and on-line ILFN classifier is efficient for fault classification and identification in machine condition monitoring.

Several qualitative issues of the ILFN classifier remain to be investigated in the near future. The important issues include

1) convergence analysis of the incremental learning rule;

TABLE VIII
PERCENT CORRECT CLASSIFICATION OF THE ILFN CLASSIFIER FOR THE WESTLAND VIBRATION DATA WITH DIFFERENT TORQUE LEVELS FOR TRAINING AND TESTING

Testing data (torque levels)	Training data (torque levels)									
	100%	80%	75%	70%	60%	50%	45%	40%	27%	all torque levels
100%	100	60.57	35	36	40	40	50	30.8	40	100
80%	71.43	100	71.29	59.71	50	33.33	36.36	33.33	33.33	100
75%	81.42	71.43	100	96.29	40	33.33	36.36	33.33	39	100
70%	57.14	71.43	100	100	80	33.33	36.36	33.33	39	100
60%	41.2	48.6	80.4	93.2	100	92.6	56	40	78.2	100
50%	37.2	59.8	80	80	99.8	100	100	100	100	100
45%	42.5	50	50	53.5	81.25	100	100	100	95.75	100
40%	4	40	60	59.3	80.2	100	100	100	100	100
27%	7.4	42.4	60	60	80	80.6	91.25	100	100	100
Hidden nodes	8	7	7	7	5	5	4	5	5	51

Remark: (1) Only 10 patterns were used for training when the same torque load was used for testing.
(2) All patterns were used for training when a different torque load was used for testing.
(3) For the last column, only 10 patterns from each torque load level were used for training.

TABLE IX
PERCENT CORRECT CLASSIFICATION OF THE MLP, RFBN, LVQ, AND ILFN NETWORK, TRAINED BY 100% TORQUE LEVEL, AND TESTED BY DIFFERENT TORQUE LEVELS

Testing data (torque level)	Classifier types (trained with 100% torque level)			
	MLP	RBFN	LVQ	ILFN
100%	96.5	100	100	100
80%	58.71	4.57	71.43	71.43
75%	61.14	0.57	74.29	74
70%	37.43	0.57	57.14	57.44
Learning type	OFF-LINE	OFF-LINE	OFF-LINE	ON-LINE
Learning time	400 Sec 475 epochs	12 Sec 1 epoch	194 Sec 500 epochs	4 Sec 1 epoch

TABLE X
THE ILFN CLASSIFIER ASSIGNED CLASSES TO THE UNSEEN PATTERNS

Learned Fault	Faults	Labeled classes			
	No fault	0	0	0	0
Unseen faults	Fault 8	0	0	0	1
	Fault 5	0	0	1	0
	Fault 2	0	0	1	1
	Fault 3	0	1	0	0
	Fault 6	0	1	0	1
	Fault 7	0	1	1	0
	Fault 4	0	1	0	0

2) parameter survey of the initial standard deviation σ_0, and the threshold ε;
3) generalization performance of the ILFN classifier with respect to the number of hidden neurons.

REFERENCES

[1] G. G. Yen, "Health monitoring of vibration signatures in rotorcraft wings," *Neural Process. Lett.*, vol. 4, no. 3, pp. 127–137, 1996.

[2] ——, "Autonomous neural control in flexible space structures," *Contr. Eng. Practice*, vol. 3, no. 4, pp. 471–483, 1995.

[3] A. C. McCormick and A. K. Nandi, "Classification of the rotating machine condition using artificial neural networks," in *Proc. Inst. Mech. Eng. C*, vol. 211, 1997, pp. 439–450.

[4] W. J. Staszewski and K. Worden, "Classification of faults in gearboxes preprocessing algorithms and neural networks," *Neural Comput. Applicat.*, vol. 5, no. 3, pp. 160–183, 1997.

[5] R. H. Cabell, C. R. Fuller, and W. F. Obrien, "Neural-network modeling of oscillatory loads and fatigue damage estimation of helicopter components," *J. Sound Vibration*, vol. 209, no. 2, pp. 329–342, 1998.

[6] P. Fu, A. D. Hope, and M. A. Javed, "Fuzzy classification of milling tool wear," *Insight*, vol. 39, no. 8, pp. 553–557, 1997.

[7] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. New York: Academic, 1990.

[8] M. Nadler and E. P. Smith, *Pattern Recognition Engineering*. New York: Wiley, 1992.

[9] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Netw.*, vol. 3, pp. 698–713, Oct. 1992.

[10] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[11] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing*. Cambridge, MA: MIT Press, 1986.

[12] T. Kohonen, *Self-Organizing Maps*, 2nd ed. New York: Springer-Verlag, 1997.

[13] D. S. Broomhead and D. Lowe, "Multivariable function interpolation and adaptive networks," *Complex Syst.*, vol. 2, pp. 321–355, 1988.

[14] S. Haykin, *Neural Networks: A Comprehensive Foundation*. New York: MacMillan, 1994.

[15] Y. S. Hwang and S. Y. Bang, "An efficient method to construct a radial basis function neural network classifier," *Neural Netw.*, vol. 10, no. 8, pp. 1495–1503, 1997.

[16] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Comput.*, vol. 1, no. 2, pp. 281–294, 1989.

[17] L. Zadeh, "Fuzzy sets," *Inform. Contr.*, vol. 8, pp. 338–353, 1965.

[18] H. Ishibuchi, K. Nozaki, and H. Tanaka, "Distributed representation of fuzzy rules and its application to pattern classification," *Fuzzy Sets Syst.*, vol. 52, pp. 21–32, 1992.

[19] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.

[20] J. C. Bezdek, S. K. Chuah, and D. Leep, "Generalized k-nearest neighbor rules," *Fuzzy Sets Syst.*, vol. 18, no. 3, pp. 237–256, 1986.

[21] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, pp. 580–585, Apr. 1985.

[22] R. L. Chang and T. Pavlidis, "Fuzzy decision tree algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 28–35, Jan. 1977.

[23] S. Mitra, R. K. De, and S. K. Pal, "Knowledge-based fuzzy MLP for classification and rule generation," *IEEE Trans. Neural Netw.*, vol. 8, pp. 1338–1350, Dec. 1997.

[24] V. Uebele, S. Abe, and M. S. Lan, "A neural-network-based fuzzy classifier," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 353–361, Feb. 1996.

[25] P. Vuorimaa, T. Jukarainen, and E. Karpanoja, "A neuro-fuzzy system for chemical agent detection," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 415–424, Aug. 1995.

[26] J. S. Jang, "ANFIS: Adaptive-network-based fuzzy inference systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 665–685, Mar. 1993.

[27] C. F. Juang and C. T. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 12–32, Feb. 1998.

[28] P. K. Simpson, "Fuzzy min–max neural networks—Part 1: Classification," *IEEE Trans. Neural Netw.*, vol. 3, pp. 776–786, Oct. 1992.

[29] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Netw.*, vol. 4, pp. 759–771, 1991.

[30] J. R. Williamson, "Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps," *Neural Netw.*, vol. 9, no. 5, pp. 881–897, 1996.

[31] G. Tontini and A. A. de Queiroz, "RBF fuzzy-ARTMAP: A new fuzzy neural network for robust on-line learning and identification of patterns," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, vol. 2, 1996, pp. 1364–1369.

[32] N. B. Karayiannis and G. W. Mi, "Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques," *IEEE Trans. Neural Netw.*, vol. 8, pp. 1492–1506, Dec. 1997.

[33] R. A. Fisher, "The use of multiple measurements in axonomic problems," *Ann. Eugenics*, vol. 7, pp. 179–188, 1936.

[34] D. H. Deterding, "Speaker normalization for automatic speech recognition," Ph.D. dissertation, Mass. Inst. Technol., Dept. Elec. Eng. Comput. Sci., Cambridge, 1989.

[35] M. White. (1995, Feb.) CMU neural networks benchmark collection. Carnegie Mellon University's School of Computer Science. [Online]. Available: http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/neural/bench/cmu/0.html.

[36] B. G. Cameron, "Final report on ch-46 aft transmission seeded fault testing," Westland Helicopters, Westland Res. Paper RP907, Sept. 1993.

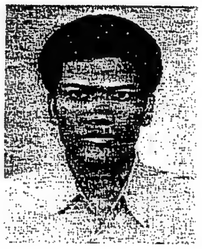
[37] A. J. Robinson, "Dynamic error propagation networks," Dept. Engineering, Mass. Inst. Technol., Cambridge, 1989.

[38] D. A. Linkens and J. Nie, "Learning control using fuzzified self-organizing radial basis function neural network," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 280–287, Aug. 1993.

Gary G. Yen (S'87–M'88–SM'97) received the Ph.D. degree in electrical and computer engineering from the University of Notre Dame, Notre Dame, IN, in 1992.

He is currently an Associate Professor in the School of Electrical and Computer Engineering, Oklahoma State University (OSU), Stillwater. Before he joined OSU in 1997, he was with the Structure Control Division, U.S. Air Force Research Laboratory, Albuquerque, NM. His research is supported by the DoD, DoE, EPA, NASA, NSF, and the process industry. His research interest includes intelligent control, computational intelligence, conditional health monitoring, signal processing, and their industrial/defense applications.

Dr. Yen was an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS and *IEEE Control Systems Magazine* from 1994 to 1999. He served as Publicity Chair for the 1997 IEEE Conference on Decision and Control, San Diego, CA, as Finance Chair for the 1998 IEEE International Symposium on Intelligent Control, Gaithersburg, MD, as Finance Chair for the 1999 IEEE Conference on Control Applications, and as Program Chair for the 2000 IEEE Conference on Control Applications, Anchorage, AK. On behalf of IEEE Robotic and Automation Society and later IEEE Control Systems Society, he was a Representative to the IEEE Neural Network Council Administrating Committee. He is currently serving as Chair for IEEE Control Systems Society Student Activities Standing Committee and Chair for IEEE Neural Network Council Neural Network Technical Committee.



Phayung Meesad (S'97) received the B.S. degree in electrical engineering from King Mongkut's Institute of Technology, North Bangkok, Thailand, in 1994, and the M.S. degree in electrical engineering from Oklahoma State University, Stillwater, in 1998, where he is currently pursuing the Ph.D. degree in electrical engineering.

His research interests include neural networks, fuzzy systems, fuzzy neural networks, genetic algorithms, pattern classification, control systems, signal processing, and machine health monitoring.

Haber, R. and Unbehauen, H., "Structure identification of nonlinear dynamic systems- a survey on input/output approaches," *Automatica*, **26**, pp. 651-677, 1990.

He, X. and Asada, H., "A new method for identifying orders of input-output models for nonlinear dynamic systems," *Proceeding of American Control Conference*, San Francisco, California, pp. 2520-2523, 1993.

Jordan, M. I. and Jacobs, R. A., "Hierarchical mixtures of experts and the EM algorithm," *Neural Computation*, **6**, pp. 181-214, 1994.

Kadirkamanathan, V. and Fabri, S. G., "Recursive structure estimation for nonlinear identification with modular networks," *Proceedings of IEEE Workshop on Neural Networks for Signal Processing VIII*, Cambridge, England, pp. 343-350, 1998.

Kaplan, D. and Glass, L., *Understanding Nonlinear Dynamics*, Spinger-Verlag: New York, NY, 1995.

Malti, R., Ekongolo, S. B. and Ragot, J., "Dynamic SISO and MISO system approximations based on optimal Laguerre models," *IEEE Transactions on Automatic Control*, **43**, pp. 1318-1323, 1998.

Murray-Smith, R. and Johansen, T. A., *Multiple Model Approaches to Modelling and Control*, Taylor & Francis: London, UK, 1997.

Narendra, K. S., "Neural networks for control: theory and practice," *Proceedings of the IEEE*, **84**, pp. 1385-1405, 1996.

Narendra, K. S., Balakrishnan, J. and Ciliz, K. M., "Adaptation and learning using multiple models, switching, and tuning," *IEEE Control Systems Magazine*, **15**, pp. 37-51, 1995.

Narendra, K. S. and Mukhopadhyay, S., "Adaptive control using neural networks and approximate models," *IEEE Transactions on Neural Networks*, **8**, pp. 475-485, 1997.

Narendra, K. S. and Parthasarathy, K., "Identification and control of dynamic systems using neural networks," *IEEE Transactions on Neural Networks*, **1**, pp. 4-27, 1990.

Nelles, O., Nonlinear System Identification with Local Linear Neuro-Fuzzy Models, *Technical University of Darmstadt, Institute of Automatic Control*, Germany, 1998.

Oliveira e Silva, T., "On the determination of the optimal pole position of Laguerre filters," *IEEE Transactions on Signal Processing*, **43**, pp. 2079-2087, 1995.

Pinkus, A., *N-Widths in Approximation Theory*, Springer-Verlag, New York, NY, 1985.

Pomerleau, D., "Reliability estimation for neural network based autonomous driving," *Robotics and Autonomous Systems*, **12**, pp. 104-113, 1994.

Principe, J. C., Wang, L. and Motter, M. A., "Local dynamic modeling with self-organizing maps and applications to nonlinear system identification and control," *Proceedings of the IEEE*, **86**, pp. 2240-2258, 1998.

Stenman, A., Model on Demand: Algorithms, Analysis and Application, Ph.D. Dissertation, *Linkoping University*, Sweden, 1990.

Tøffner-Clausen, S., *System Ientification and Robust Control, A Case Study Approach*, Springer-Verlag: New York, NY, 1996.

Wahlberg, B., "Laguerre and Kautz Models," *Proceedings of IFAC Symposium on System Identification*, Copenhagen, Denmark, pp. 965-976, 1994.

APPENDIX H:

Adaptive Critic Fault Tolerant Control Using Dual Heuristic Programming

by

Gary G. Yen and Pedro de Lima

Submitted to
IEEE Transactions on Neural Networks

Adaptive Critic Fault Tolerant Control Using Dual Heuristic Programming

Gary G. Yen and Pedro G. DeLima

Abstract – **A hierarchical architecture that combines a high degree of reconfigurability and long-term memory is proposed as a fault tolerant control algorithm for complex nonlinear systems. Dual Heuristic Programming (DHP) is used for adapting to faults as they occur for the first time in an effort to prevent the build up of a general failure, and also as a tuning device after switching to a known scenario. A dynamical database, initialized with as much information of the plant as available, oversees the DHP controller. The decisions of which models to record, when to intervene and where to switch are autonomously taken based on specifically designed quality indexes. The results of the application of the complete algorithm to a proof-of-the-concept numerical example help to illustrate the fine interrelations between each of its subsystems.**

Index Terms – **Fault tolerant control, fault detection and isolation, neural networks, adaptive critic, multiple models.**

I. Introduction

Increased performance requirements are often achieved at the cost of plant and control simplicity. As overall complexity rises, so does the chance of occurrence, diversity and severity of faults. Therefore, availability, defined as the probability that a system or equipment will operate satisfactorily and effectively at any point of time [1], becomes a factor of great importance. For automated production processes, for example, availability is now considered to be the single factor with the highest impact on profitability [2]. Fault Tolerant Control (FTC) is a field of research that emerges to increase availability and reduce the risk of safety hazards by specifically designing control algorithms capable of maintaining stability and performance despite the occurrence of faults [3,4].

As complex systems suffer from faults, the original model parameters, or even their own dynamic structure, may change in a multitude of unpredictable ways. Even if the system has a satisfactory linearization around the nominal operating point, nonlinearities may become of paramount importance after a fault occurs [5]. Since complex systems pose a challenge even in the design of models under nominal conditions, the task of off-line devising nonlinear high order models for all known fault scenarios can be a daunting one. When the stochastic nature of faults is taken into consideration, and to even possess knowledge of all fault scenarios is made impossible, it becomes clear to see that the problem of interest to FTC cannot be dealt with without on-line nonlinear adaptive control strategies. In the proposed architecture, Dual Heuristic Programming (DHP), an Adaptive Critic Design (ACD), was chosen as the reconfigurable controller due to its known effectiveness to work in noisy, nonlinear environments while making minimal assumptions regarding the nature of that environment [6].

To our best knowledge, the application of the DHP reconfigurable controller represents one of the most effective ways to deal with the unexpected dynamics that a plant may assume after the occurrence of a fault. However, as a FTC scheme by itself, the use of a reconfigurable controller such as DHP presents two main limitations. The first one arises from the fact that solutions to a set of expected fault scenarios are often available and may involve the application of very specific control laws. A reconfigurable controller alone however, does not provide any mechanism through which knowledge available during design time can be incorporated. The second limitation arises from the known tradeoff between adaptation and long-term memory. As the reconfigurable controller provides faster convergence to a wider range of control solutions, it fails to retain the knowledge of the control laws designed for previously visited scenarios.

To overcome both limitations, a novel supervisor system oversees the DHP controller in the architecture displayed in Figure 1. The Identifier and Controller Dynamical Database (ICDD), located inside the supervisor contains the knowledge available during design time, as well as solutions devised online for unexpected fault scenarios. The decisions of when to intervene by switching to a known control solution and when to add a new identifier and controller pair to the database are taken by the supervisor based on the current fault scenario. Such information is extracted by the scenario recognition module, which makes use of specifically designed quality indexes capable, not only of performing Fault Detection and Identification (FDI), but also to produce indispensable information on the evolution of a fault through time. The synergetic combination of the superior adaptation capabilities of the DHP controller with the fault information and long-term memory provided by the multiple model structure [7] of the proposed supervisor generates an advanced FTC scheme capable to deal with a diversified collection of actuator and component faults.

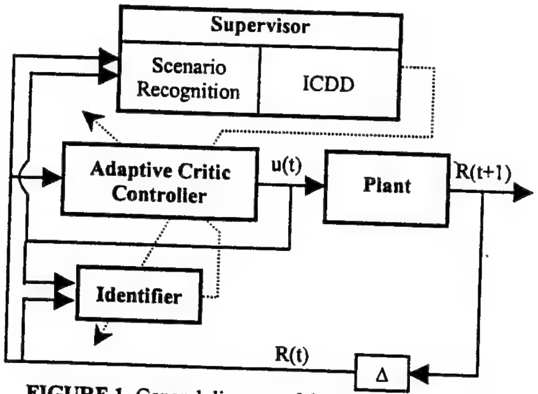


FIGURE 1. General diagram of the proposed scheme.

The remaining of the letter is organized as follows. In Section II, the problem is stated along with a description of the goal of the FTC application, followed by the presentation of the reconfigurable controller in Section III. In Section IV the proposed supervisor scheme is described, and its interaction with the controller is discussed in detail. Finally, Section V brings a numerical example through which some of the key features of the proposed FTC are illustrated. Section VI concludes the letter with pertinent observations.

II. Problem Statement

The present work focuses on complex nonlinear systems upon which a certain control law was applied to make it capable of performing its mission under nominal operational conditions. The system under nominal conditions is thereby considered to be stable and with the desirable degree of performance. Due to the generic nature of the considered faults, no guarantee of stability in a fault scenario is required of the nominal control law.

This letter focuses on actuator and component faults, leaving sensor faults to be identified and recovered in parallel by any of the currently available methods such as sensor fusion [8,9] and specialized filters [10]. For a recovery to be at all possible, the required redundancy (hardware or analytical) is supposed to exist in the system. From the theoretical point of view, this statement matches the requirement for sustained observability and controllability through fault scenarios.

As stated in the IFAC – SAFEPROCESS terminology definition [3], the goal of fault tolerant control is to maintain control objectives, although a loss of performance may be accepted. Even though speed is not an imperative issue, the system must converge in a finite time to a stable trajectory as close to the desired one as possible under the effects of certain faults. Stability during transient phases is also a key feature to achieve a safe and efficient fault accommodation scheme. Trajectory tracking and time to recover are both quantitative measures that can be used to evaluate the performance of FTC methods.

Noise and low intensity disturbances are always present in real world applications, requiring robust control solutions. Stronger disturbances and component aging are examples of situations that may cause significant changes in the system. In the proposed scheme, those changes may reflect in variation of parameters of the identified model, requiring the controller to be self adaptive. In extreme cases, FTC requires the controller to completely restructure itself to cope with drastic changes in the dynamics of the plant.

III. Reconfigurable Controller Using Adaptive Critic Design

In order to achieve the required degrees of reconfiguration and stability, the adaptive controller can benefit greatly if more than the simple instantaneous difference between desired and actual states is available to be used as a performance index. Due to the continuous interaction between the controller and the plant however, the quality of a certain control strategy can only be fully measured after analyzing all future effects it has on the control mission, which in our case is trajectory tracking. Such measure of control quality is translated into the Hamilton-Jacobi-Bellman Equation (1) that defines the cost-to-go $J(t)$ [11].

$$J(t)=\sum_{k=0}^{\infty}\gamma^{k}U(t+k), \qquad (1)$$

where γ is a discount factor for finite horizon ($0<\gamma<1$), and $U(t)$ is the (primary) utility function or local cost that can be defined as the squared tracking error at each instant. This kind of problem formulation is the main focus of dynamic programming, a machine learning technique that solves it through a backward search from the final step [12]. To make the problem tractable to an on-line learning approach, Heuristic Dynamic Programming (HDP), an ACD, introduces a critic block responsible for approximating the cost-to-go $J(t)$ [13]. The controller, usually referred to as the actor or action block in ACD literature [14], is then adapted in the direction of the minimization of $J(t)$ by using the critic to analyze the impact of its actions over the cost-to-go. DHP is a more advanced ACD capable of providing superior performance due to smoother and more accurate information on how $J(t)$ is affected by the control signals by using the critic to directly estimate the partial derivative of the cost-to-go with respect to the plant states.

Although ACD's can be implemented with any differentiable structure [12], neural networks have been widely used [15,16] due to their generalization and nonlinear mapping capabilities as well as having suitable methods for on-line learning [17]. Given the system of interest, Recurrent Neural Networks (RNN) were chosen due to their more efficient handling of dynamic nonlinear mapping [18]. Backpropagation Through Time (BPTT) [19] was used to extract the approximation of the partial derivatives required during training.

IV. Supervisor

To better understand the functionality of the supervisory system, it can be divided into three layers. The first layer collects and analyzes data from the plant and the controller block. The second one is responsible for the decision making, while the final layer devises ways to implement the resolutions.

The first layer receives the sampled output of the plant $R(t)$ and a delayed input $u(t-1)$, computes two quality indexes and indicates the known scenario that better approximates the current dynamics. The first quality index, $q_c(t)$, measures the reconfigurable controller performance by performing a decaying sum of the primary utility function of the active controller as shown in Equation (2),

$$q_c(t)=\int_{0}^{t}e^{-\xi_c(t-\tau)}U(\tau)d\tau, \qquad (2)$$

where $0<\xi_c<1$ is a time decay factor.

For the calculation of the second quality index, the delayed input is then fed into the model database. The database contains a copy of the identification, action and critic networks used to control the plant under nominal operation as well as copies for each known fault scenario. Each one of the identification networks in the database is then used to generate an identification error. For each one of those, a decaying sum similar to (2) is used, and the results are compared. As shown in (3), the smallest identification error history defines $q_i(t)$, the identification quality index, and the model d that generates it is appointed as the switching candidate.

$$q_i(t)=\min_{d\in D}\left(\int_{0}^{t}e^{-\xi_i(t-\tau)}\left|R^{d}(\tau)-R(\tau)\right|d\tau\right), \qquad (3)$$

where $0<\xi_i<1$ is a time decay factor and $R^{d}(t)$ is the vector of states predicted by the identified d, which in turn is an element of the set of identifiers in the database D.

In the second layer, a threshold is defined for each of the quality indexes, dividing them into high (Hq_c, Hq_i) and low (Lq_c, Lq_i) values. The threshold for $q_c(t)$ defines what is to be considered as an acceptable performance, while the one for $q_i(t)$ stipulates the degree of similarity of the input-output behavior that should be used to consider two models distinct. Four states, tagged 1 to 4, are in this way defined, and the decision process illustrated in Figure 2 takes place. It's important to notice that in this formulation the actions of switching and adding to the database take place in the transition between states. This characteristic, added to the improved smoothness of the quality indexes bestowed by the regressive mean, aids in the generation of the hysteresis required to prevent the automatic switching scheme to generate spurious oscillations between states.

If both indexes are low (state 1), the current reconfigurable controller is performing satisfactorily over a known environment, and no action is required. While in this state, an abrupt fault may cause the performance to be degraded enough for the controller quality index $q_c(t)$ to surpass its threshold. In this case, $q_i(t)$ will remain low or grow on the respective events of a known or unknown fault.

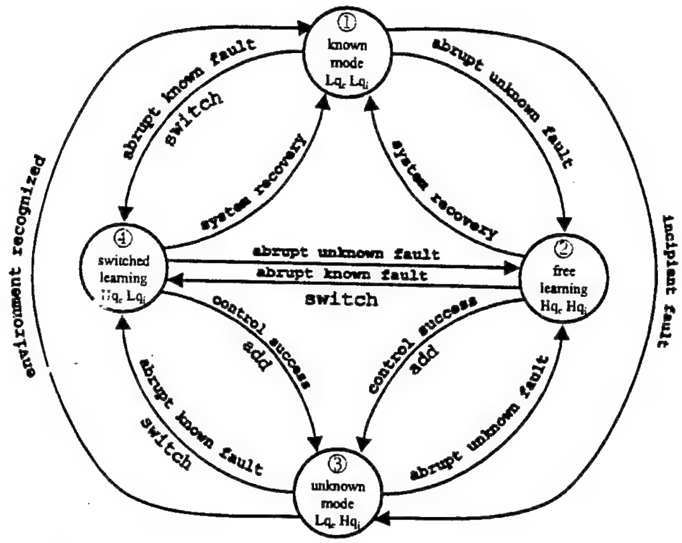


FIGURE 2. Decision graph of the second layer of the supervisory system. The states, tagged 1 to 4, are defined by the quality measures $q_c(t)$ and $q_i(t)$. The moments when the actions of switching and adding to the database are performed are shown on the graph.

If both indexes exceed the threshold (state 2), the environment has abruptly changed due to an unknown fault, and the supervisor is unable to provide any help to the DHP controller. If $q_i(t)$ remains low (state 4), there is already a set of DHP parameters in the ICDD previously adapted to deal with a plant with similar dynamics and to which switching should take place. The decision process then remains in state 4 (Hq_c and Lq_i) until either the system is recovered or another fault takes place before that. If the composite fault is also a known fault, switching takes place again triggered by the change in the switching candidate appointed by the first decision layer.

Incipient faults, often connected to component aging, may be gradually adapted by the reconfigurable controller and eventually indicate a high $q_i(t)$, even though $q_c(t)$ remains low during all the process (transition from state 1 to 3). In this case, there is no purpose in learning a new environment/controller pair since the parameters are continuously changing. As a matter of fact, if allowed to learn all the transient models, the database might rapidly grow to an intractable size.

When the DHP controller is adapting to a new environment (state 2), $q_c(t)$ is expected to decrease to the point where it crosses its threshold (transition to state 3), and a new set of parameters is added to the ICDD, but two other scenarios must also be considered. The first one deals with the possibility of an abrupt known fault to happen before the first fault is completely dealt with. In this case $q_i(t)$ reaches a low value prior to $q_c(t)$ and switching to the known environment takes place. The second scenario addresses the situation in which, due to actuator or physical limitations, although the controller is capable of reconfiguring itself to generate a stable trajectory, the performance remains below the desired level. In such case, the decision logic remains in state 2 as the reconfigurable controller cannot be improved by supervisor intervention.

The third layer manipulates the ICDD by making new entries and by switching to the reconfigurable controller indicated by the first layer, when requests arrive from the second. Switching is implemented by loading a complete set of parameters of the three neural networks (i.e. identification, action and critic networks) to the DHP algorithm currently being used. The fact that the controller is switched to one devised to a similar plant and the natural generalization capabilities of neural networks add to improved stability when the parameters are loaded as new initial conditions to the adaptive process. The database also stored copies of all the partial derivatives required when updating the networks using backpropagation through time. Uploading those derivatives also works to increase switching smoothness since more information about the new dynamics of the plant is supplied.

V. Numerical Example

In this section, a numerical example is used as an illustration of the dynamics of the proposed FTC algorithm. Special emphasis is given to the actions of the supervisor system, which is the intelligent core of the algorithm. For the sake of simplicity and understanding, the plant consists of a simple linear ARMA model, subject to faults resulting in changes in its parameters. This example by no means reflects any limitations of the architecture, which has been designed to deal with complex nonlinear plants that may have their very dynamics altered by the occurrence of a fault. The models, sampled at 10Hz, used to simulate the plant under nominal operation and under each of the artificial faults, are given below.

<u>Nominal</u> $$G_n(s)=\frac{1.25}{s^2+2.0s+1.0}$$

<u>Fault 1</u> $$G_{f1}(s)=\frac{-0.108s+4.95}{s^2+36.9s+9.5}$$

<u>Fault 2</u> $$G_{f2}(s)=\frac{0.004s+0.99}{s^2+2.0s+1.0}$$

<u>Fault 3</u> $$G_{f3}(s)=\frac{-0.349s-5.41}{s^2+6.3s+2.5}$$

<u>Incipient</u> $$G_{inc}(s)=\frac{0.001s+1}{s^2+1.9s+2.1}$$

The incipient fault occurs over the nominal model, changing its dynamics gradually until the one given above. The simulation was carried with the plant being abruptly changed to a different model at every 10 minutes. The goal is to follow a trajectory composed by a sine wave that changes the amplitude randomly at every half a period. Since in this case no beforehand information about the system is given to the plant, initially the nominal plant is treated as an unknown fault and therefore, as displayed in Figure 3, the system begins in the state where both quality indexes are high.

After the initial transient response, as soon as $q_c(t)$ indicates a low value, the supervisor flags a control success and adds the nominal model to the ICDD. The copy of the identification network, that is now part of the ICDD, generates a low identification error causing $q_i(t)$ to drop sharply.

The identification quality index $q_i(t)$ remains low after a model is added even though the training has been stopped and new inputs are being supplied, indicating two main achievements of the proposed FTC scheme. The first one is that the neural network used as an identifier in the DHP architecture was capable of converging to represent the true dynamics of the system. The second is that the supervisor was able to recognize the proper moment when a new identifier and controller pair should be memorized.

After the first 10 minutes of simulation the first fault occurs abruptly, changing the dynamics of the plant. While the reconfigurable controller adapts itself to the new scenario, both indexes grow, indicating that the system is going through an abrupt unknown fault. As $q_c(t)$ drops to an acceptable level, the first failure mode is recorded along with the controller that was specifically designed on-line to deal with it.

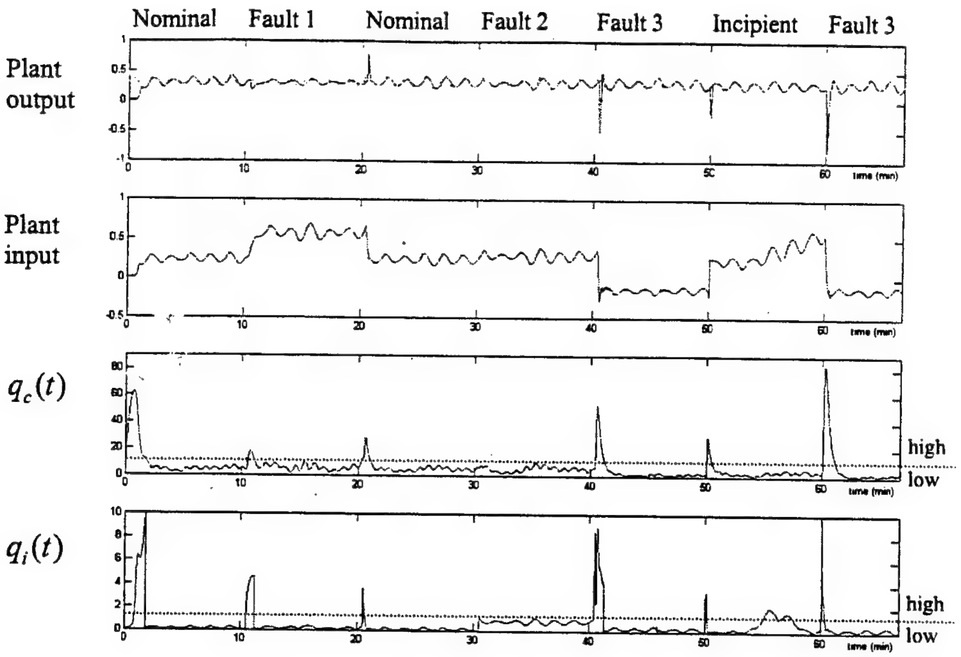


FIGURE 3. The top graph brings the desired trajectory (dashed), the output of the plant (solid) and the output of the identification network while it adapts (dotted). The second graph displays the input to the plant as calculated by the adaptive critic controller. The third and fourth graphs show the quality indexes $q_c(t)$ and $q_i(t)$ respectively, along with the thresholds used.

After 20 minutes of simulation, the plant returns to the nominal mode. Due to the change in the dynamics, $q_c(t)$ increases due to the drop in the performance. On the other hand, $q_i(t)$ shows only a thin spike indicating that there already exists an element in the ICDD that was previously designed to deal with a system similar (in this case identical) to the present one. Therefore switching takes place leading to a much faster response.

The second fault is introduced at 30 minutes. By comparing with the identifier adapted for the nominal plant, the supervisor concludes that the dynamics are not different enough to justify a new entry in the database. This property is of extreme importance in order to achieve a database capable of covering all the known space, while maintaining a compact set of recorded models. The third fault on the other hand, requires a major reconfiguration in the controller, and so it is also added to the ICDD after convergence.

After 50 minutes of simulation, the plant is instantly reverted to the nominal model, and the incipient fault is applied over it. Since in the initial moments the dynamics are still similar enough to the ones of the nominal model, switching takes place moments after 50 minutes. As the parameters of the plant are changing, the controller is capable of constantly reconfiguring itself, and the tracking error remains low. As the dynamics of the plant get more different from the nominal ones, $q_i(t)$ increases to the point when the supervisor correctly diagnoses the occurrence of an incipient fault. Around 57 minutes, $q_i(t)$ once again falls as the input-output relation of the plant becomes increasingly similar to the one stored when the first fault was learned.

To illustrate the effectiveness of the algorithm when a fault presents itself for the second time, fault 3 is introduced again at 60 minutes. As soon as the environment is recognized as a known one by low values of $q_i(t)$, switching takes place generating a smoother and faster response.

VI. Conclusion

A multiple model approach to FTC based on an intelligent dynamic database was presented. The application of DHP as a reconfigurable controller was shown to give the hierarchical algorithm the amount of flexibility required to deal with both abrupt and incipient changes in the plant. The supervisor system was used to accelerate convergence of the method by loading new initial conditions to the DHP when the plant is affected by a known abrupt fault. A methodology was presented through which new fault scenarios are recognized and assimilated on-line by the database along with parameters for the corresponding controller. Finally, these properties were successfully illustrated in the in-depth exploration of the numerical simulation example. Although the results so far have been greatly encouraging, formal investigation of online stability and real-world complications is required and will be carried out in future research.

References

1. K. Åström, P. Albertos, M. Blanke, A. Isidori, W. Schaufelberger and R. Sanz (Eds.), *Control of Complex Systems*, Springer, London, UK, 2001.
2. M. Blanke, R. Izadi-Zamanabadi, S. Bøgh and C. Lunau, "Fault-tolerant control systems – a holistic view," *Control Engineering Practice*, vol. 5, no. 5, pp. 693-702, 1997.
3. M. Blanke, M. Staroswiecki and N. E. Wu, "Concepts and methods in fault-tolerant control," *Proc. American Control Conference*, pp. 2606-2620, 2001.

4. G. Yen and L. Ho, "Intelligent on-line fault tolerant control for unanticipated catastrophic failures," *Proc. American Control Conference*, pp. 4204-4208, 2000.
5. Y. Diao, "Fault tolerant systems design using adaptive estimation and control," PhD dissertation, *Ohio State University*, Columbus, OH, 2000.
6. D. Prokhorov, R. Santiago and D. Wunsch, "Adaptive critic designs: a case study for neurocontrol," *Neural Networks*, vol. 8, no. 9, pp. 1367-1372, 1995.
7. K. Narendra and J. Balakrishnan, "Adaptive control using multiple models," *IEEE Trans. Automatic Control*, vol. 42, no. 2, pp. 171-187, 1997.
8. M. Blanke, R. Izadi-Zamanabadi and T. Lootsma, "Fault monitoring and re-configurable control for a ship propulsion plant," *Int. Journal of Adaptive Control and Signal Processing*, vol. 12, pp. 671-688, 1998.
9. G. Yen and W. Feng, "Winner take all expert network for sensor fusion," *ISA Trans.*, vol. 40, no. 2, pp. 99-110, 2001.
10. S. Qin and W. Li, "Detection and identification of faulty sensors with maximized sensitivity," *Proc. American Control Conference*, pp. 613-617, 1999.
11. J. Murray, C. Cox, R. Saeks and G. Lendaris, "Globally convergent approximate dynamic programming applied to an autolander," *Proc. American Control Conference*, pp. 2901-2906, 2001.
12. D. Prokhorov and D. Wunsch, "Adaptive critic designs," *IEEE Trans. Neural Networks*, vol. 8, no. 5, pp. 997-1007, 1997.
13. G. Lendaris and L. Schultz, "Controller design (from scratch) using approximate dynamic programming," *Proc. International Symposium on Intelligent Control*, pp. 31-36, 2000.
14. G. Venayagamoorthy, R. Harley and D. Wunsch, "Comparison of a heuristic programming and a dual heuristic programming based adaptive critics neurocontroller for a turbogenerator," *Proc. International Joint Conference on Neural Networks*, vol. 3, pp. 233-238, 2000.
15. P. Werbos, "Stable adaptive control using new critic designs," available at xxx.lanl.gov/abs/adap-org/9810001, 1998.
16. L. Feldkamp, G. Puskorius and D. Prokhorov, "Unified formulation for training recurrent networks with derivative adaptive critics," *Proc. International Conference on Neural Networks*, pp. 2268-2272, 1997.
17. K. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 4-27, 1990.
18. E. Sanchez and M. Bernal, "Adaptive recurrent neural control for nonlinear systems tracking," *IEEE Trans. Systems, Man and Cybernetics*, vol. 30, no. 6, pp. 886-889, 2000.
19. P. Werbos, "Backpropagation though time: what it does and how to do it," *Proc. of the IEEE*, vol. 78, no. 10, pp. 1550-1560, 1990.

APPENDIX I:

Rank-Density Based Multiobjective Genetic Algorithm and Benchmark Test Function Study

by

Haiming Lu and Gary G. Yen

Submitted to
IEEE Transactions on Evolutionary Computation

RANK-DENSITY BASED MULTIOBJECTIVE GENETIC ALGORITHM AND BENCHMARK TEST FUNCTION STUDY[*]

Haiming Lu   Gary G. Yen
Intelligent Systems and Control Laboratory
School of Electrical and Computer Engineering
Oklahoma State University, Stillwater, OK, 74078

Abstract— Since the 1980's, the application of Evolutionary Algorithms (EA's) in solving Multiobjective Optimization Problems (MOPs) has been receiving a growing interest from evolutionary computation community. To search for a family of "acceptable" solutions, a so called Pareto set, by using EA's population-based parallel searching ability, several MultiObjective Evolutionary Algorithms (MOEAs) have been proposed. However, most of these MOEAs have difficulty in dealing with the trade-off between uniformly distributing the computational resources and finding the *near-complete* and *near-optimal* Pareto set. On the other hand, according to the No Free Lunch theorems, no formal assurance of an algorithm's general effectiveness exists if insufficient knowledge of the problem characteristics is incorporated into the algorithm domain. In this paper, the authors propose a new evolutionary approach to multiobjective optimization problems, the Rank-Density based Genetic Algorithm (RDGA) that synergistically integrates selected features from existing MOEAs in a unique way. A new ranking method, automatic accumulated ranking strategy, and a "forbidden region" concept are introduced, completed by a revised adaptive cell density evaluation scheme and a rank-density based fitness assignment technique. In addition, four types of MOP features, such as discontinuous and concave Pareto front, local optimality, high-dimensional decision space and high-dimensional objective space are exploited and the corresponding MOP test functions are designed. By examining the selected performance indicators, RDGA is found to be statistically competitive with four state-of-the-art MOEAs in terms of keeping the diversity of the individuals along the trade-off surface, tending to extend the Pareto front to new areas and finding a well-approximated Pareto optimal front.

1. INTRODUCTION

In many scientific and engineering disciplines, it is not uncommon to face a design challenge when there are several criteria or design objectives to be met simultaneously. If these objectives are conflicting, then the problem becomes one of finding the best possible designs that satisfy the competing objectives under different trade-off scenarios. With these multiple objectives and constraints taken into consideration, an optimum design problem can then be formulated. This type of problems is known as *multiobjective*, *multicriteria* or *vector optimization* problems [1].

[*] This work was supported in part by the U.S. Air Force Office of Scientific Research under Grant F49620-98-1-0049 and National Science Foundation under I/UCRC Measurement and Control Engineering Center.

Multiobjective Optimization (MO) is a very demanding research topic because most real-world problems have not only a multiobjective nature, but also many open issues to be answered qualitatively and quantitatively. In fact, there is not even a universally accepted definition of "optimum" as in single-objective optimization [2], because the solution to an MOP is generally more than a single point. It consists of a family of points in a feasible solution space, which describes the trade-off characters among contradicted objectives.

A formal notion of Pareto optimality is given by Fonseca and Fleming in [3]. Consider, without loss of generality, the minimization of the n components $f_k, k = 1, \cdots, n,$ of a vector function $\mathbf{f}$ of a decision vector $\mathbf{x}$ in a universe U, where $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \cdots, f_n(\mathbf{x}))$. Then a decision vector $\mathbf{x}_u \in U$ is said to be Pareto-optimal if and only if there is no $\mathbf{x}_v \in U$ for which $\mathbf{v} = \mathbf{f}(\mathbf{x}_v) = (v_1, \cdots, v_n)$ *dominates* $\mathbf{u} = \mathbf{f}(\mathbf{x}_u) = (u_1, \cdots, u_n)$, that is, there is no $\mathbf{x}_v$ for such that

$$\forall i \in \{1, \cdots, n\}, v_i \le u_i, \qquad \text{and} \quad \exists i \in \{1, \cdots, n\}, v_i < u_i \; . \tag{1}$$

The set of all Pareto-optimal decision vectors is referred to as the *Pareto-optimal* set of the problem. The corresponding set of objective vectors is called the non-dominated set, or *Pareto front.* Apparently, the Pareto front dominates all other possible solutions, and in many cases, it is located on the boundary of the objective space (i.e., feasible solution space) as shown in Figure 1 for a two-objective optimization problem.

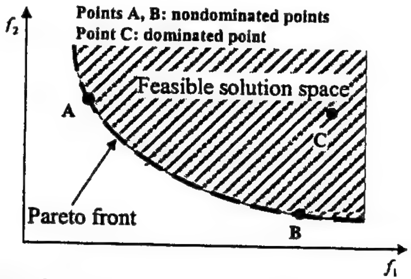


Figure 1 Graphical illustration of the Pareto optimality of a two-objective minimization problem

In their early development, Evolutionary Algorithms (EA's), a class of population-based optimization approaches, have been recognized to be well suited for multiobjective optimization. In EA's, multiple individuals search for multiple solutions in parallel, advantageously producing a family of feasible solutions to the problem. The ability to handle complex problems involving features such as discontinuity, multimodality and disjoint objective spaces, reinforces the potential effectiveness of EA's in multiobjective search and optimization, which is perhaps *the* problem area where EA's most distinguish themselves from the other alternatives [3].

Since the 1980's, several Multiobjective Evolutionary Algorithms (MOEAs) have been proposed and applied in Multiobjective Optimization Problems (MOPs) [1]. These algorithms share the same purpose— searching for a *uniformly distributed*, *near-optimal* and *near-complete* Pareto front for a given MOP. However, this ultimate goal is far from being accomplished by the existing MOEAs described in literature. In one respect, most of the MOPs are very complicated and require the computational resources to be homogenously distributed in a high-dimensional search space. On the other hand, those better-fit individuals generally have strong tendencies to restrict searching efforts within local areas because of the "genetic drift" phenomenon [4-5], which results into the loss of diversity due to stochastic sampling. This phenomenon is a well-known trade-off decision pertaining to the *efficiency* and *efficacy* dilemma [6].

Additionally, to show or judge these MOEAs' performances, most of the researchers used numeric MOPs as the benchmark problems. In MOEA community, limited benchmark test functions are frequently exploited in publications, because it makes the performance comparison of different algorithms relatively easy and straightforward. However, based on the No Free Lunch (NFL) theorems [7], no formal assurance of an algorithm's general effectiveness exists if insufficient knowledge of the problem domain is incorporated into the algorithm design. Therefore, visually comparing MOEA performances on non-standard and unjustified numeric MOPs does little to determine a given MOEA's actual efficiency and effectiveness. A standard set of carefully designed benchmark test functions exhibiting relevant MOP domain characteristics can provide the necessary fair comparative basis [8].

In this paper, the authors conduct an extensive study on four selective MOP features that may produce difficulties for MOEAs to search for true and uniformly distributed Pareto fronts. Function ***F1*** is originated from an existing MOP to create a discontinuous and concave Pareto front. Functions ***F2-1*** and ***F2-2*** are designed to explore local and global Pareto optimality caused by objective functions and constraints, respectively. Functions ***F3*** and ***F4*** are test functions to show the complications for MOEAs in coping with MOPs involving high-dimensional decision and objective spaces, respectively. Four representative MOEAs (Fonseca's MOGA [9], PAES [10], SPEA II [11] and NSGA-II [12]) were applied to generate the simulation results for each test function. In addition, a new multiobjective optimization approach, Rank-Density based Genetic Algorithm (RDGA), proposed in this paper is also examined by these test functions for a comparative study. We show that RDGA synergistically integrates selected features of existing MOEAs in a unique way and has advantages over the other algorithms under consideration in finding a *near-optimal*, *near-complete* and *uniformly distributed* Pareto front. The simulation results show the proposed RDGA is competitive with the selective MOEAs measured by some performance metrics. In this paper, we make no distinction between MOEAs and MOGAs, because their names are mostly chosen for historical reasons. Different approaches have been "recombined", which makes a classification no longer feasible. For instance, the Pareto Archived Evolutionary Strategy (PAES)

[10] uses binary representation from GA, while Non-dominated Sorting Genetic Algorithm II (NSGA-II) [12] uses a "plus" selection (elitist) from evolutionary strategies.

The remainder of this paper is organized as follows. Section 2 reviews existing literature on the most well regarded MOEAs. Section 3 proposes a new Rank-Density based Genetic Algorithm that is designed to deal with high-dimensional objective functions, explore the optimality of the candidate Pareto points and maintain the diversity of the final Pareto front. In Section 4, we discuss some typical features a Pareto front may possess and how to measure the performance quantitatively. Section 5 presents four MOP benchmark functions that create discontinuous and concave Pareto front, local optimality, high-dimensional decision space and high-dimensional objective space. RDGA with four representative MOEAs are exploited to produce simulation results for the given test functions. The performance indicators of the resulting Pareto fronts are examined to compare the performances of RDGA and the chosen algorithms. Finally, Section 6 provides some concluding remarks along with pertinent observations.

2. EVOLUTIONARY MULTIOBJECTIVE OPTIMIZATION ALGORITHMS

Generally, the approximation of the Pareto-optimal set involves two conflicting objectives: the distance to the true Pareto front is to be minimized, while the diversity of the generated solutions is to be maximized [11]. To address the first objective, a Pareto based fitness assignment method is usually designed in many existing MOEAs [3] in order to guide the search toward the true Pareto optimal front. For the second objective, some successful MOEAs provide density estimation methods to preserve the population diversity. In addition, several other techniques have also been adopted such as elitism scheme [10]-[12], crowded comparison [9, 12], archive truncation [11], and etc. These methods and techniques can be found in four state-of-the-art MOEAs— MOGA, PAES, NSGA-II and SPEA II, which are briefly reviewed in the following.

2.1 Multiobjective Genetic Algorithm— MOGA

In their MOGA [9], Fonseca and Fleming proposed a ranking scheme in which the rank of a certain individual corresponds to the number of individuals in the current population by which it is dominated. Based on this scheme, all the non-dominated individuals are assigned rank 1, while dominated ones are penalized according to the population density of the corresponding region of the trade-off surface. Moreover, to prevent premature convergence of the population, a niche-formation method to distribute the population over the Pareto front in the objective space is adopted. More discussions pertaining to the ranking method in MOGA are given in Subsection 3.1.

2.2 Pareto Archive Evolutionary Strategy— PAES

As a local search algorithm that simulates a random mutation hill-climbing strategy, PAES may represent the simplest possible, yet effective, nontrivial algorithm capable of generating diverse solutions in the Pareto optimal set [10]. In PAES, a pure mutation operation is adopted to fulfill a local search scheme. A reference archive of previously found non-dominated solutions is updated at each generation in order to identify the dominance ranking of all the resulting solutions. Although (1+1)-PAES is originated as the simplest version, PAES can also generate λ mutants by mutating one of the μ current solutions, which is called ($\mu+\lambda$)-PAES [10]. Since PAES does not perform a population-based search, only tournament selection can be applied to determine the survivors of the next generation.

2.3 Non-dominated Sorting Genetic Algorithm II— NSGA-II

NSGA-II [12] was advanced from its origin, NSGA [13]. In NSGA-II, a non-dominated sorting approach is used for each individual to create Pareto rank, and a crowding distance assignment method is applied to implement density estimation. In a fitness assignment between two individuals, NSGA-II prefers the point with a lower rank value, or the point located in a region with fewer number of points if both of the points belong to the same front. Therefore, by combining a fast non-dominated sorting approach, an elitism scheme and a parameter-less sharing method with the original NSGA, NSGA-II claims to produce a better spread of solutions in some testing problems [12].

2.4 Strength Pareto Evolutionary Algorithm II— SPEA II

Similar to NSGA-II, SPEA II [11] is an enhanced version of SPEA [1]. In SPEA II, instead of calculating standard Pareto rank, each individual in both main population and elitist archive is assigned a strength value, which incorporates both dominated and density information. On the basis of the strength value, the final rank value is determined by the summation of the strengths of the individuals that dominate the current one. Meanwhile, a *k*th nearest neighbor density estimation method is applied to obtain the density value of each individual. The final fitness value is the sum of rank and density values. In addition, a truncation method is used in elitist archive in order to maintain the number of individuals contained in the archive to be constant. In the experimental results, SPEA II shows better performance than SPEA [1] over all the test functions considered therein.

3. RANK-DENSITY BASED GENETIC ALGORITHM

From the literature review, the main deficiency in the existing MOEAs lies on designing a suitable fitness assignment strategy in order to search for a *near-complete* and *near-optimal* approximated Pareto front for the given optimization problem. Unfortunately, these two objectives are contradictory. In one respect, the "genetic drift" character of EA needs to be exploited to converge the solution to a near optimal point. On the other hand, the "genetic drift" phenomenon must be avoided in order to sketch a uniformly sampled trade-off surface for the final Pareto front. Based on these considerations, a new Rank-Density based Genetic Algorithm (RDGA), which converts a high-dimensional MOP into a bi-objective optimization problem to minimize fitness rank values and cell densities, is proposed. Six crucial procedures of RDGA will be discussed as follows.

3.1 Automatic Accumulated Ranking Strategy (AARS)

First introduced by Goldberg [14], and applied by NSGA-II [12], the pure Pareto ranking method ensures that all the non-dominated individuals in the population will be assigned rank 1 and removed from temporary assertion, then a new set of non-dominated individuals will be assigned rank 2, and so forth. Fonseca [9] further improved the ranking method by including the density information into the rank value—an individual's rank corresponds to how many individuals in the current population that dominate it. For example, consider an individual y at generation t, which is dominated by $p^{(t)}$ individuals in the current generation. Its rank value is given by,

$$rank(y,t) = 1 + p^{(t)}. \tag{2}$$

All the non-dominated individuals are assigned rank value 1, while dominated ones are penalized according to the population density of the corresponding region of the trade-off surface. In addition, as discussed in Subsection 2.4, SPEA II [11] applied the summation of the strength values and treated it as rank value to fulfill the same task. In this paper, we propose an Automatic Accumulated Ranking Strategy (AARS). In AARS, an individual's rank value is defined as the summation of the rank values of the individuals that dominate it. Assume for the same example, at generation t, individual y is dominated by $p^{(t)}$ individuals $y_1, y_2, \cdots, y_{p^{(t)}}$, whose rank values are already known as $rank(y_1,t)$, $rank(y_2,t)$, $\cdots$, $rank(y_{p^{(t)}},t)$. Its rank value can be computed by

$$rank(y,t) = 1 + \sum_{j=1}^{p^{(t)}} rank(y_j,t). \tag{3}$$

By AARS, all the non-dominated individuals are still assigned rank value 1, while dominated ones are penalized to reduce the population density and redundancy. For instance, suppose we want to minimize two objectives, f_1 and f_2, and MOEAs generate eleven individuals. Their rank values based on four ranking techniques proposed in NSGA-II, MOGA, SPEA II and AARS are illustrated in Figure 2, where each dot represents a candidate phenotype solution. Considering all the individuals located in the lower-right area, AARS provides the exact same rank values as those computed by the pure Pareto ranking method (adopted by NSGA-II [12]), since all the individuals are clearly aligned and not crowded at all. Therefore, adding extra density information (resulted from SPEA II) may not be necessary in this case. Meanwhile, AARS does impose a penalty to the dominated individuals located in the upper-left area. The reason of penalizing all the dominated individuals in this area is because there exist several non-dominated individuals that can mostly represent the dominated ones. Therefore, without increasing the population size, the population diversity will be kept by penalizing those dominated individuals by AARS.

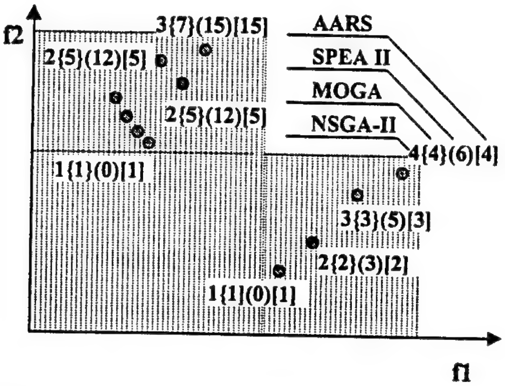


Figure 2 Individual rank values resulting from different ranking methods

3.2 Adaptive density value calculation

According to [11], although AARS and other ranking schemes [9,11] provide a sort of niching mechanism based on the concept of Pareto dominance, they may fail when most individuals do not dominate each other. Therefore, additional density information is incorporated to discriminate between individuals having identical raw fitness values. In RDGA, to deal with this problem, we adopt a modified adaptive cell density evaluation scheme originated from [10] as shown in Figure 3. The cell width in each objective dimension can be formed as

$$d_i = \frac{\max_{\mathbf{x}\in X} f_i(\mathbf{x}) - \min_{\mathbf{x}\in X} f_i(\mathbf{x})}{K_i}, i = 1,\ldots,n, \tag{4}$$

where d_i is the width of the cell in the ith dimension, K_i denotes the number of cells designated for the ith dimension (i.e., in Figure 3, $K_1 = 6$ and $K_2 = 4$), and $\mathbf{x}$ is taken from the whole decision space X. As the maximum and minimum fitness values in the objective space will change with different generations, the cell size will vary from generation to generation to maintain the accuracy of the density calculation. The density value of an individual is defined as the number of the individuals located in the same cell. Note that in PAES [10], the grid location of a solution in the objective space is obtained by repeatedly bisecting the range in each objective and finding in which half the solution is. However, RDGA uses a different scheme to locate which cell an individual belongs to. First, the cells are created by dividing the range of the current objective space based on K_i and given initial population. Second, the center position of each cell is obtained and stored. Third, each individual of the initial population searches for its nearest cell center, identifies this cell as its "home address" and considers the other individuals who share the same "home address" as its "family members". Then for each of these "homes", the number of "family members" who dwell in it are counted and saved as its density value. Fourth, when an offspring is generated and accepted, its "home address" can be easily located by following the third step and the density value of its home is increased by one. Meanwhile, if an existing individual is eliminated, its "home" is notified and the density value of its "home" is decreased by one. Therefore, at each generation, an individual can access its "home address" and then obtain the corresponding density value. The "home address" is merely a "pointer" to inform an individual where to find its density value. For instance, as shown in Figure 3, the "home address" and density value of individual A are (4,3) and 4, respectively. Therefore, if a new generated or a removed individual does not change the boundary of the range of the current objective space, only the density value of its "home" is changed. The density values of the other "homes" (cells) will not be affected. This setting can avoid the unnecessary recalculation of an unchanged range of the objective space and density values.

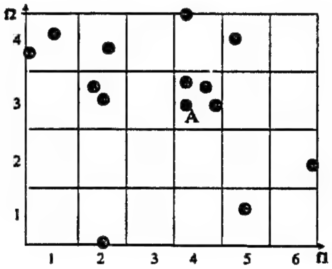


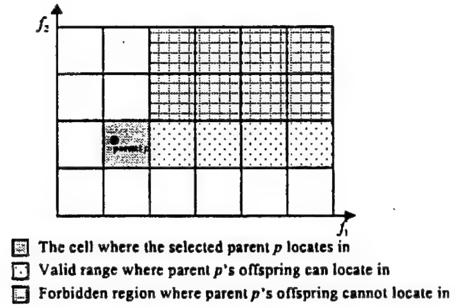
Figure 3 Density map and density grid

3.3 Rank and density based fitness assignment

Because rank and density values represent fitness and population diversity, respectively, we assigned them as two important attributes to each individual. Therefore, any multiobjective optimization problem can be converted into a bi-objective optimization problem. On the other hand, since we need to minimize rank value together with density value, some further modifications need to be made to the original notation.

First, instead of minimizing the density value of an individual, we minimize the density value of the entire population. Based upon the definition of the cell density, an individual located in a crowded cell must have a relatively higher density value, which contributes much more to the population density value than an individual in the sparse area does. For example, a cell containing ten individuals will contribute $10\times10=100$ to the population density value, whereas a cell containing only one individual will contribute 1 to the population density value. The average individual distance value can be obtained by dividing the current population density value by the current population size.

Second, after the rank and density values of each individual have been extracted, a modified Vector Evaluated Genetic Algorithm (VEGA) [15] is applied to fulfill the fitness assignment. It is well known that VEGA possesses two deficiencies: 1) it does not have a scheme to maintain the diversity of the evolved Pareto front, and 2) it has difficulty in dealing with the problems involving concave trade-off surfaces. However, as mentioned above, the goal of RDGA is to find the non-dominated individuals with the rank value equal to 1 and reduce the population density value to obtain a uniformly distributed trade-off surface. In this setting, there is no concern about keeping the population diversity in the rank-density domain. Furthermore, whether the "Pareto front" in the rank-density domain is concave is not an issue, since it is not a real Pareto front for the MOP under consideration. Therefore, a simple VEGA is effective enough to fulfill the fitness assignment after the original optimization problem has been transformed into the rank-density domain. It is worthy of noting that the idea of converting multiobjective into a domination measure function and neighboring density function was originated in [16] and later revised in [17]. However, in their papers, two newly formulated objective functions are derived from Goldberg's ranking scheme [14] and Horn's niche sharing method [18]. Afterward they combine two objective functions into one nonlinear fitness function, which is considered as the final fitness function. Because rank and density values have distinct characteristics, it is very difficult for this algorithm to designate a suitable coefficient in *ad hoc* to bias the preference during the evolutionary process.



f_2
parent p
f_1
▨ The cell where the selected parent *p* locates in
▤ Valid range where parent *p*'s offspring can locate in
▦ Forbidden region where parent *p*'s offspring cannot locate in

Figure 4 Illustration of the valid range and the forbidden region

3.4 Crossover and mutation

For crossover, the parent selection and replacement schemes are borrowed from Cellular GA [19] to explore the new search area by "diffusion." For each subpopulation, a fixed number of parents are randomly selected for crossover. Then, each selected parent performs crossover with the best individual (the one with the lowest rank value) within the same cell and the nearest neighboring cells that contain individuals. If one offspring produces better fitness (a lower rank value or a lower population density value) than its corresponding parent, it replaces its parent. The replacement scheme of the mutation operation is analogous. Meanwhile, as RDGA takes the minimization of the population density value as one of the objectives, it is expected that the entire population will move toward an opposite direction to the Pareto front where the population density value is being minimized. Although moving away from the true Pareto front can reduce the population density value, obviously, these individuals are harmful to the population to converge to the Pareto front. To prevent "harmful" offspring from surviving and affecting the evolutionary direction and speed, a *forbidden region* concept is proposed in the replacement scheme for the density subpopulation, thereby preventing the "backward" effect. The *forbidden region* includes all the cells dominated by the selected parent. The offspring located in the forbidden region will not survive in the next generation, and thus the selected parent will not be replaced. As shown in Figure 4, suppose our goal is to minimize objectives f_1 and f_2, and a resulting offspring of the selected parent p is located in the forbidden region. By RDGA, this offspring will be eliminated even if it reduces the population density, because this kind of offspring has the tendency to push the entire population away from the desired evolutionary direction.

As discussed in Subsection 3.1, Automatic Accumulative Ranking Strategy (AARS) includes the scheme of punishing the individuals located in crowded areas, which means we add a bias to avoid the population density value from expanding too much when RDGA is implementing the minimization of the rank value. Meanwhile, a forbidden region is brought in to introduce another bias to prevent the offspring from having higher ranks than their parents when RDGA is evolving a lower population density value. Therefore, RDGA can be interpreted as trying to convert an MOP into two new single objective optimization problems— minimizing rank and density values, and then performing an evolutionary process to optimize each of the objectives in turn. It is necessary to note that these two biases make two objectives of RDGA highly correlated. When one objective is being optimized, the corresponding bias will take the other objective as a constraint to keep the computation resources homogeneously distributed between two objectives.

3.5 Constraint handling

To handle the constraints, every new generated offspring will be tested against all the constraint functions in order to determine if it is a valid solution. If the offspring satisfies all the constraints, it will be evaluated by the fitness function to obtain its fitness value, otherwise, it will be discarded.

3.6 Archiving the candidate Pareto points

The elitism scheme in [20] is also adopted in RDGA. At each generation, the non-dominated individuals created from the main population will be copied and stored to an archive. Meanwhile, a non-dominated solution in the archive may also be selected with a certain probability as a parent to perform genetic operation. This probability p_e^t is called "elitism intensity" and according to [1], at each generation t, the probability of sampling an individual from the archive is given by

$$p_e^t = 1 - \left(\frac{|B|}{|A| + |B|}\right)^2, \qquad (5)$$

where A and B represents the archive of elitists and main population, respectively. After the evolution process has terminated, the resulting solutions in both the main population and archive will be compared to derive the final Pareto front.

4. MOP TEST FUNCTION— FEATURES AND INDICATORS

According to [21], in order to compare the performance of different MOEAs, the design of a variety of MOP benchmark problems and performance metrics is essential. Because a multiobjective optimization problem can be closely related to a combination of Single objective Optimization Problems (SOPs), some literature review on the features of SOP test functions can be helpful. In De Jong's SOP test bed study [22], he declared that six problem characteristics need to be examined: continuous and discontinuous, convex and nonconvex, unimodal and multimodal, quadratic and nonquadratic, low and high dimensionality, and deterministic and stochastic. In addition, Michalewicz [23] addressed other issues that need to be considered for the SOP test bed design, such as the number of constraints, type of constraints and ratio between the feasible and complete search space. Apparently, some of these properties are also valuable for an MOP and must be incorporated into the test bed design. Nevertheless, because the purpose of solving an MOP is to search for a *near-complete* set of non-dominated solutions (Pareto front), the features that cause the true Pareto front difficult to be found are the primary concerns in the MOP test function design. By now, in MOEA community, several well-crafted benchmark test functions have been designed and applied by different researchers [24-27]. In this paper, we focus our investigation on four distinct types of MOP test functions. They are MOPs with discontinuous and concave Pareto front, global/local optimality, high-dimensional decision space and high-dimensional objective space.

We deploy five MOEAs— MOGA, PAES, NSGA-II, SPEA II and the proposed RDGA in the simulation and run each of the algorithms 50 times to obtain the statistical results. For each run, a new initial population with 100 individuals is randomly generated and used by each of the four population-based MOEAs (i.e., MOGA, NSGA-II, SPEA II and RDGA), while only one initial individual is generated for PAES according to its design procedure [10]. The archive size is set to be 100 for all the selective MOEAs that involve elitism scheme. We use three indicators derived from the final generation of 50 runs to benchmark the comparison results via statistical Box plots. They are average individual rank value, average individual density value and average individual distance. As discussed in Subsection 3.1, for an individual, different ranking schemes will produce different rank values, which will be used in respective fitness evaluations and selections. However, for a fair comparison in terms of ranking indicators for different MOEAs, we use Goldberg's pure Pareto ranking method [14] to recalculate the rank value for each individual resulted from each applied MOEAs. Meanwhile, as shown in Figure 3, the average individual density value is calculated as the mean value of all the individual density values over the entire population. Here, according to the population size, we choose the number of grids for each objective dimension to be 20. Furthermore, because the rank is a relative value, it must be stated that we cannot guarantee the final population will be a true Pareto set, even if all of its individuals have rank values equal to 1 as shown in Figure 5. For this reason, we use "final individual distance" as the third indicator to show how far the non-dominated points on the resulting final Pareto front PF_{final} are away from the true Pareto front PF_{true}, where PF_{true} is known in *a priori* for the given test functions in this paper. This indicator was originally introduced by Veldhuizen and Lamont [28], where the final individual distance G is defined as

$$G = \frac{(\sum_{i=1}^{m} d_i^2)^{1/2}}{m}, \tag{6}$$

where m is the number of individuals in PF_{final}, and d_i is the Euclidean distance between each of these individuals and a point on PF_{true} that is the closest to it. A result of $G = 0$ indicates the convergence $PF_{final} = PF_{true}$; any other value indicates PF_{final} deviates from PF_{true}.

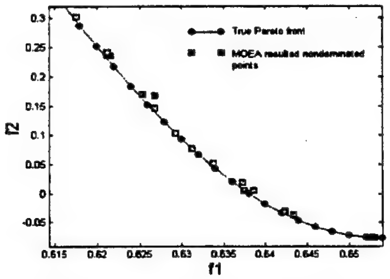


Figure 5 Difference between PF_{true} and PF_{final}

Moreover, in order to compare the dominance relationship between two populations resulted from two different MOEAs, the coverage of two sets (C value) [1] is measured to show how the final population of one algorithm dominates the final population of another algorithm. Function C maps the ordered pair (X_i, X_j) to the interval [0, 1], where X_i and X_j denote the final populations resulted from algorithm *i* and *j*, respectively. The value $C(X_i, X_j) = 1$ implies that all points in X_j are dominated by or equal to points in X_i. The opposite, $C(X_i, X_j) = 0$, represents the situation when none of the points in X_j are covered by the set X_i. Note that both $C(X_i, X_j)$ and $C(X_j, X_i)$ need to be considered independently since they have the distinct meanings.

Therefore, four indicators represent qualitative measures that describe the quality of the final result of selected MOEAs— the average individual rank value shows the dominated relationship between different individuals, the average individual density value illustrates how good the population diversity is preserved, the average individual distance measures distance between PF_{final} and PF_{true}, which provides the quality of the resulting Pareto front, and the C value compares the domination relationship of a pair of MOEAs. All the values of four performance indicators generated at the final generation are illustrated by Box plots to derive the statistical comparison results.

5. MOP Test Function Case Study

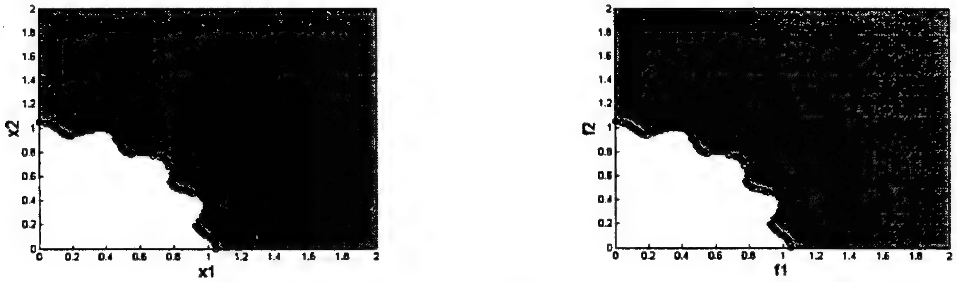
To examine the performances of the selected MOEAs and the proposed RDGA on the test functions with different Pareto front features, we explore four numerical test functions in the simulation study. Function *F1* is advanced from an existing MOP to create a discontinuous and concave Pareto front [29]. Functions *F2-1* and *F2-2* are designed to explore local and global Pareto optimality caused by objective functions and constraints, respectively. Function *F3* has a high-dimensional decision space, while function *F4* involves a high-dimensional objective space. For a fair comparison, the stopping generation, the chromosome length of each decision variable, the crossover rate and the mutation rate are chosen to be 10,000, 15, 0.7 and 0.1, respectively for all population-based MOEAs considered. A one point crossover is used for all the population based MOEAs. In addition, we select (1+10)-PAES and a bit flip mutation rate $1/k$ is used for a chromosome of k genes. The tournament size t_{dom} defined in [10] is chosen to be 2.

5.1 *F1*— MOP with discontinuous and concave Pareto front

The rationale of exploiting MOPs with discontinuous and concave Pareto fronts is that some MOEAs using plain aggregating schemes have been proven of having difficulty in finding the Pareto points on the discontinuous and concave segments. MOEA's ability of finding a nonconvex Pareto front is one of the most important reasons of using EA's other than traditional gradient-based or simplex-based algorithms in multiobjective optimization.

In this paper, a modified Tanaka's MOP [29] is chosen to be the test function with a discontinuous and concave Pareto front.

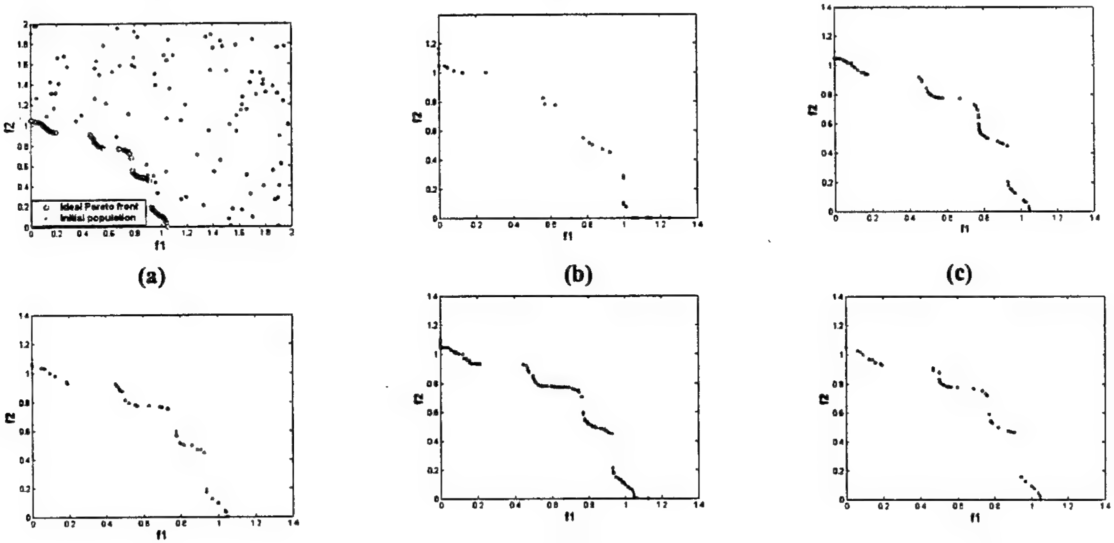
$$\boxed{\begin{array}{l} \text{Minimize } f_1(x_1,x_2) \text{ and } f_2(x_1,x_2)\text{, where} \\ \qquad f_1(x_1,x_2)=x_1 \\ \qquad f_1(x_1,x_2)=x_2 \\ \text{subject to } 0 \le x_1,x_2 \le \pi\ ,(x_1-0.5)^2-5(x_2-0.5)^2<0, \\ \qquad -(x_1^2+x_2^2)+1+0.1\cos(16\arctan(\frac{x_1}{x_2}))\le 0\,. \end{array}} \qquad (7)$$



(a) (b)

Figure 6(a) Decision space and Pareto optimal set (b) Objective Space and Pareto front of function *F1*

Indeed, the concave feature is created by the complicated constraints imposed in Equation (7). The Pareto optimal set and the true Pareto front are the same for this problem since each objective variable is equal to one decision variable. Figure 6(a) shows the Pareto optimal set, and Figure 6(b) shows the corresponding Pareto front, which includes five discontinuous segments and all of them possess concavity features. Figure 7(a) shows the true Pareto front and a randomly generated initial population. Using the same initial population for all population-based MOEAs, Figures 7(b)–(f) present the resulting Pareto fronts by five MOEAs. The Box plots for the average values of three indicators over 50 runs are illustrated in Figures 8(a), (b) and (c), respectively. The performance measures of $C(X_i, X_j)$ for the comparison sets between algorithms i and j are shown in Figure 9, where algorithms 1 – 5 represent MOGA, NSGA-II, PAES, RDGA and SPEA II in alphabetical order, respectively.



(a)

(b)

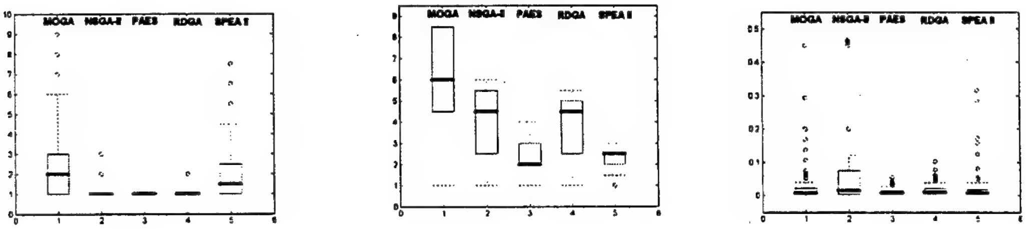
(c)

(d)

(e)

(f)

Figure 7(a) Ideal Pareto front and a randomly generated initial population; (b) – (f) Pareto fronts resulted from (b) MOGA; (c) NSGA-II; (d) PAES; (e) RDGA; and (f) SPEA II on test function *F1*



(a)

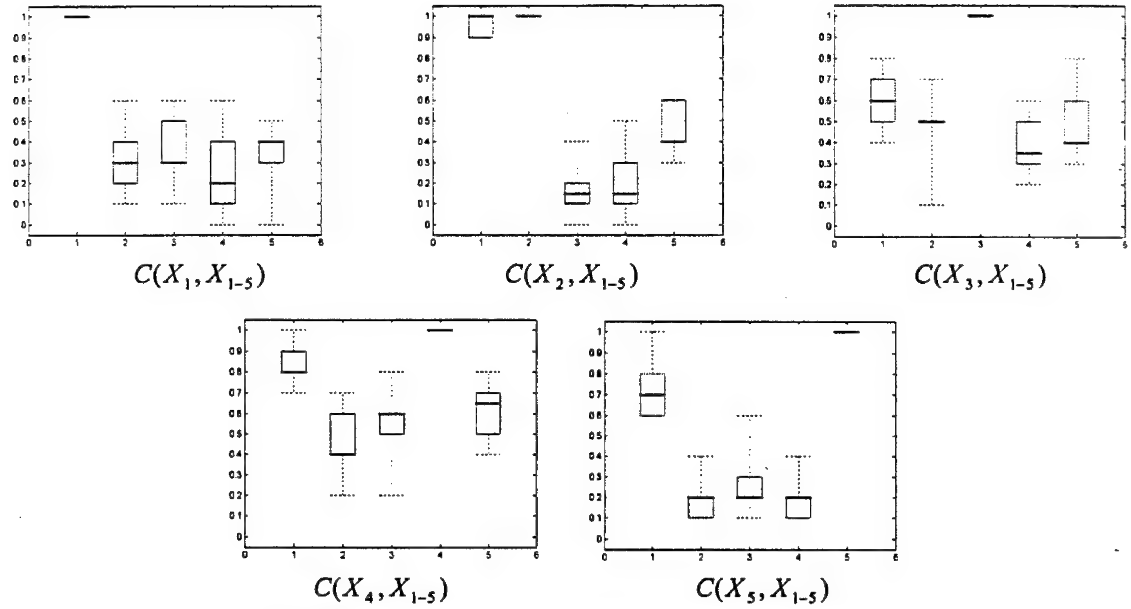
(b)

(c)

Figure 8 Box plots of (a) average individual rank value; (b) average individual density value; and (c) average individual distance on test function *F1*

Apparently, comparing the resulting Pareto fronts and indicator values in Figures 7– 9, we can see that MOGA has the lowest performance in terms of all the indicator values, while the other four MOEAs provide competitive results. In particular, RDGA produces more complete Pareto fronts than the other four MOEAs, and it also provides the highest $C(X_4, X_{1\sim5})$ values, which means the solution set resulted from

RDGA most likely dominate the rest of the solution sets resulted from the other selective MOEAs. However, it is worthy to mention that the solution set resulted from RDGA also has relatively high density and distance values, which can be explained as RDGA creates more Pareto points than the other MOEAs and some of these points are not true non-dominated points. This problem can be solved if we let RDGA run a longer time instead of the predetermined 10,000 generations.



$C(X_1, X_{1\text{-}5})$ $C(X_2, X_{1\text{-}5})$ $C(X_3, X_{1\text{-}5})$

$C(X_4, X_{1\text{-}5})$ $C(X_5, X_{1\text{-}5})$

Figure 9 Box plots based on C measure on test function *F1*

5.2 *F2*— Local and global Pareto optimality

Deb [30] proposed a multimodal two-objective optimization problem that possesses a local and a global Pareto front. He suggested that MOEAs might have a great tendency to converge to the local Pareto front instead of the global Pareto front if a certain kind of initial population was used. However, he did not elaborate the detail of the design procedure and how to make the problem more challenging. Moreover, a further study is needed if the local optimality is caused by constraints instead of objective functions, because two different rules behind each of them may result in dissimilar effects.

5.2.1 *F2-1*— Local optimality resulted by objective function

A two-variable, two-objective local-Pareto testing problem with a local Pareto front can be designed as:

$$\boxed{\begin{array}{l} \text{Minimize } f_1(x_1,x_2) \text{ and } f_2(x_1,x_2)\text{, where} \\ \qquad f_1(x_1,x_2)=R(x_1,x_2) \\ \qquad f_2(x_1,x_2)=\dfrac{T(x_1,x_2)}{S(x_1,x_2)} \\ \text{where } T(x_1,x_2)=A-p_1\times e^{\frac{(x_1-y_1)^2}{q_1}}-p_2\times e^{\frac{(x_2-y_2)^2}{q_2}}, \\ \text{subject to } C(x_1,x_2). \end{array}} \qquad (8)$$

From Equation (8), we can see in $T(x_2, x_2)$, parameter A affects the lowest bound of the feasible solution space and Pareto front; p_1 and p_2 determine the optimality of y_1 and y_2. If $p_1 > p_2$, y_1 will be the global optimal point, and y_2 will be the local optimal point. Otherwise, y_2 will be the global optimum, and y_1 will be the local optimum. Meanwhile, the deviation between y_1 and y_2 determines the distance of the gap between local and global optima. Parameters q_1 and q_2 determine how sharp the curves around the optimal points y_1 and y_2 will be. If $q_1 << q_2$, a global optimal point is created with a spike around y_1, and the sharper the spike is, the thinner the global Pareto optimal set will be.

A test function ***F2-1*** is created from the general model in Equation (8) as:

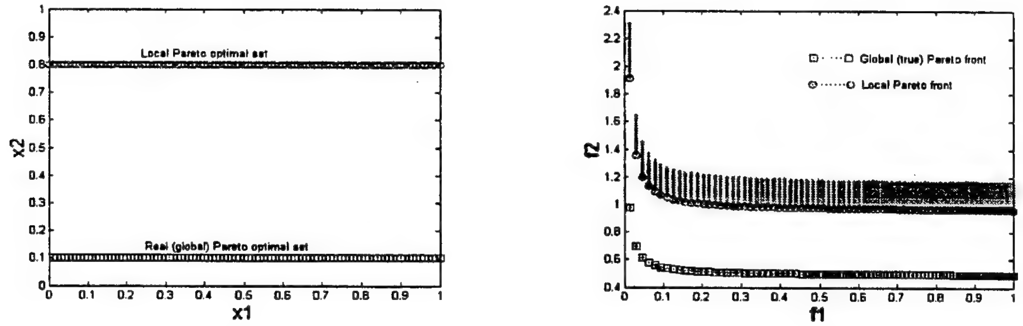
Minimize $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$, where

$$f_1(x_1, x_2) = \sin(\frac{\pi}{2} x_1)$$

$$f_2(x_1, x_2) = \frac{(1 - e^{-\frac{(x_2 - 0.1)^2}{0.0001}}) + (1 - 0.5e^{-\frac{(x_2 - 0.8)^2}{0.8}})}{\arctan(100 x_1)} \quad (9)$$

subject to $0 \le x_1, x_2 \le 1$.

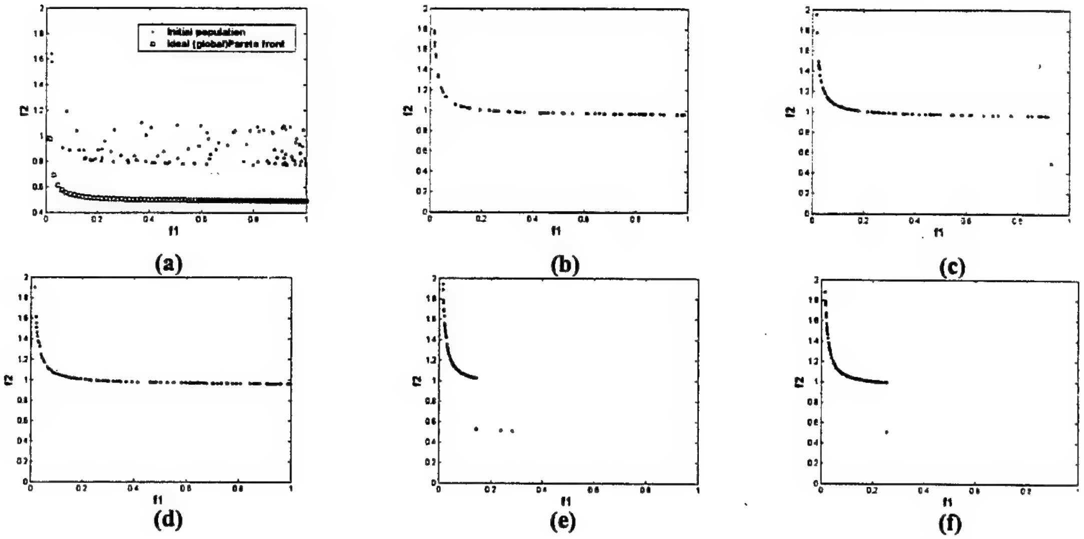
In Equation (9), there are two optimal values for x_2, $x_{2_{global}} = 0.1$ and $x_{2_{local}} = 0.8$, which are global optimum and local optimum for $f_2(x_1, x_2)$, respectively. This effect will construct the final local and global Pareto fronts as shown in Figure 10(b) with a sampling rate equal to 0.01 for both decision variables. The true (global) Pareto front is a very thin curve, which is separated from the major range that contains the local Pareto front.



(a) (b)

Figure 10(a) Decision space and Pareto optimal set (b) Objective space and local and global Pareto fronts of function *F2-1*

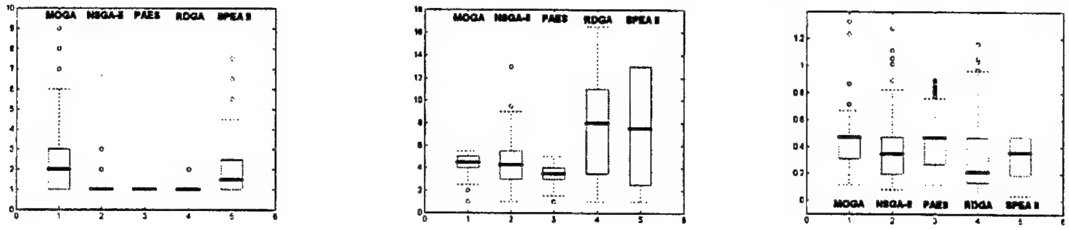
Figure 10(a) shows the decision space and local and global Pareto optimal sets, while Figure 10(b) shows the objective space and local and global Pareto fronts for the test function ***F2-1***. Figures 11(b)–(f) show resulting Pareto fronts by five chosen MOEAs for a randomly generated initial population, which is plotted in Figure 11(a) with a true Pareto front. The Box plots for the average values of three indicators over 50 runs are illustrated in Figures 12(a), (b) and (c), respectively. The performance measures of $C(X_i, X_j)$ for the comparison sets between algorithms i and j are shown in Figure 13, where algorithms 1 – 5 represent MOGA, NSGA-II, PAES, RDGA and SPEA II in alphabetical order, respectively.



(a) (b) (c)

(d) (e) (f)

Figure 11(a) Ideal Pareto front and a randomly generated initial population; (b) – (f) Pareto fronts resulted from (b) MOGA; (c) NSGA-II; (d) PAES; (e) RDGA; and (f) SPEA II on test function *F2-1*



(a) (b) (c)

Figure 12 Box plots of (a) average individual rank value; (b) average individual density value; and (c) average individual distance on test function *F2-1*

From Figure 13, we can see that RDGA and SPEA II provide the best results. Particularly, RDGA's lowest C value is greater than 0.8, which means most of the solutions resulted from the other four MOEAs are dominated or equal to the solutions by RDGA. Moreover, RDGA produces the lowest rank and distance values. The highest density values generated by RDGA and SPEA II are caused by the partial local and partial global Pareto fronts as shown in Figure 11(e) and (f), which may result in a very crowded partial global segment. From Figure 11, it is obvious that the resulting Pareto front can be pure global, pure local or partial local and partial global. Indeed, the shapes of the resulting Pareto fronts significantly depend on different types of initial populations for this test function. Therefore, two sets of initial populations are used for comparison. Set 1 includes 50 initial populations where none of their individuals belong to the global Pareto front. For set 2, at least one individual is located on the global Pareto front for each of 50 initial populations.

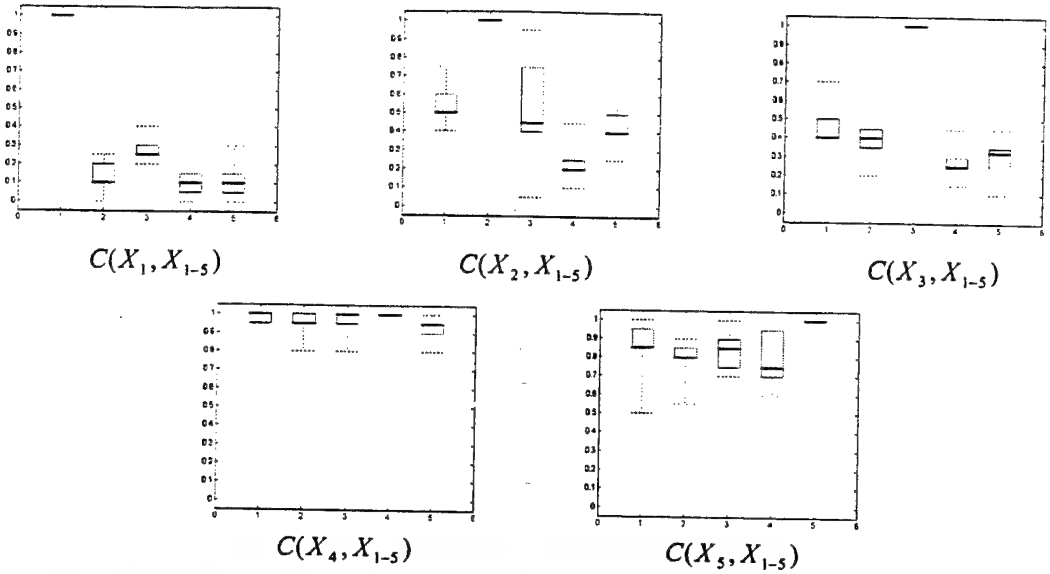


Figure 13 Box plots based on C measure on test function *F2-1*

Table 1 Final simulation results for function *F2-1* by five MOEAs using initial population set 1

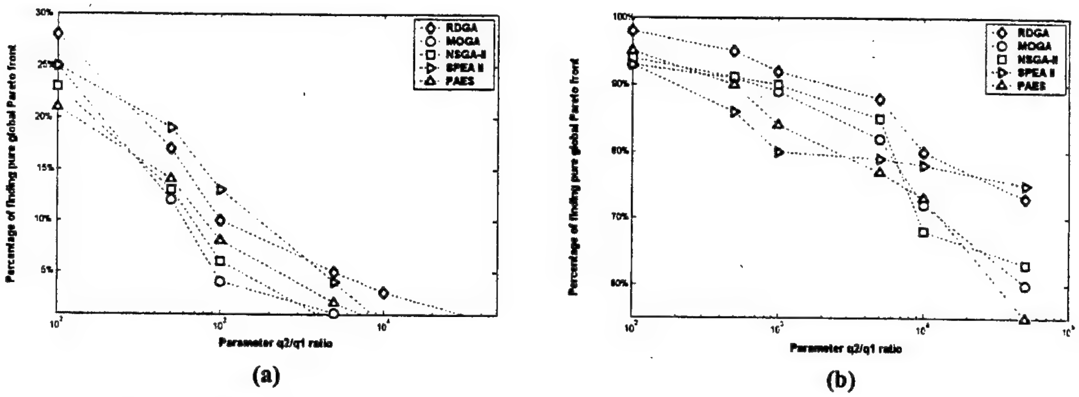
	Number of runs	Stop generation	Final average individual rank value	Final average individual density value	Final average generation distance	Number of runs produce pure global Pareto front	Number of runs produce local Pareto front*	Number of runs produce partial global Pareto front
MOGA	50	10,000	1.02	3.21	0.59	0	49	1
NSGA-II	50	10,000	1	5.03	0.51	1	45	4
PAES	50	10,000	1	3.54	0.55	0	49	1
RDGA	50	10,000	1	6.15	0.43	2	40	8
SPEA II	50	10,000	1.01	5.32	0.46	0	42	8

Table 2 Final simulation results for function *F2-1* by five MOEAs using initial population set 2

	Number of runs	Stop generation	Final average individual rank value	Final average individual density value	Final average generation distance	Number of runs produce pure global Pareto front	Number of runs produce pure local Pareto front*	Number of runs produce partial global Pareto front
MOGA	50	10,000	1.03	3.74	0.14	37	0	13
NSGA-II	50	10,000	1.03	3.30	0.05	45	0	5
PAES	50	10,000	1	4.05	0.09	41	0	9
RDGA	50	10,000	1.12	3.44	0.07	44	0	6
SPEA II	50	10,000	1.15	3.21	0.06	44	0	6

***Note: In Table 1 and 2, we consider a pseudo-global Pareto front as a local Pareto front**

Tables ·1 and 2 show the indicator values for set 1 and set 2 correspondingly. Comparing the observations from Table 1 with Table 2, we can see that all of the selected MOEAs are very sensitive to the initial population. When the initial population contains at least one individual that belongs to the global Pareto front, there will be a higher probability for the final population to converge to the global Pareto front, and otherwise it is most likely to converge to a local Pareto front. Moreover, different choices of parameters $A, p_1, p_2, q_1, q_2, y_1, y_2$ values will produce various Pareto optimality characteristics. For instance, Figures 14(a) and (b) show how parameters q_1 and q_2 affect the selected MOEAs in finding a global Pareto front for the initial population Set 1 and 2, respectively. When the ratio of q_2 / q_1 increases, the percentage that the final population is located on the global Pareto front will decrease correspondingly.



(a)

(b)

Figure 14 Illustration of q_2 / q_1 ratio affects MOEAs finding global Pareto front (a) using initial population set 1 and(b) using initial population set 2

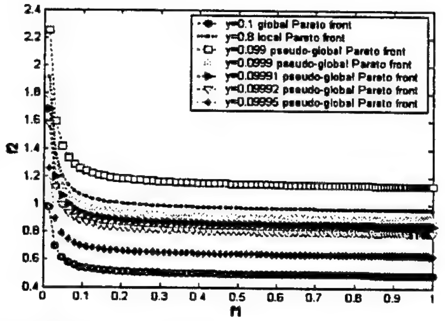


Figure 15 Pseudo-global Pareto fronts when x_2 approaches to $x_{2_{global}} = 0.1$ ($q_2 / q_1 = 10{,}000$) ratio

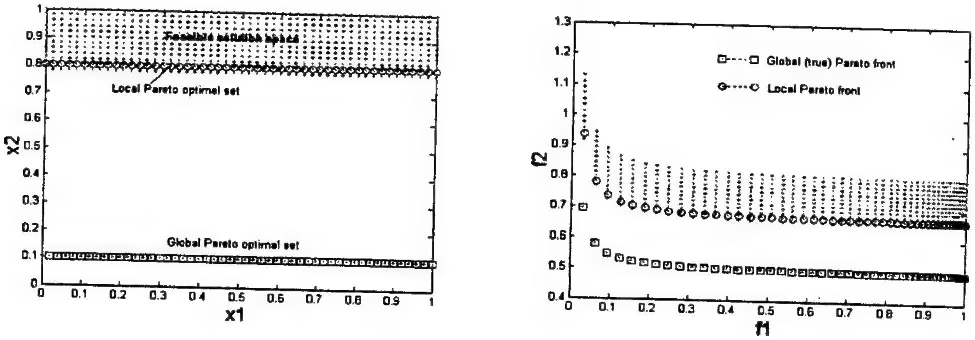
Indeed, for $q_2 / q_1 = 10{,}000$ given in Equation (9), the global Pareto optimal set is already very thin, which means there is only a very small deviation from $x_{2_{global}} = 0.1$ to produce global Pareto optimality. Even when x_2 takes a very close value to $x_{2_{global}} = 0.1$, such as $x_2 = 0.09995$, the resulting Pareto front will not be the global one, which is shown in Figure 15. From Figure 15, we also see that the gap between the local and global Pareto front is not empty. Some pseudo-global Pareto fronts will emerge when the y value is getting close to $x_{2_{global}} = 0.1$. Therefore, instead of being trapped by the local Pareto front, the resulting non-dominated points may be stuck on a pseudo-global Pareto front as well. This effect increases when the ratio of q_2 / q_1 increases. In this scenario, although RDGA may perform better than the other selected MOEAs on average, it will still be difficult to find a global Pareto front if none of the individuals of the initial population are located exactly on the global Pareto front.

5.2.2 *F2-2*—Local optimality resulted by constraint

Applying constraints may also create the similar local and global optimal effect that is represented by

$$\boxed{\begin{array}{l} \text{Minimize } f_1(x_1,x_2) \text{ and } f_2(x_1,x_2)\text{, where} \\ \qquad f_1(x_1,x_2)=\sin(\frac{\pi}{2}x_1) \\ \qquad f_2(x_1,x_2)=\dfrac{(1-e^{-\frac{(x_2-0.1)^2}{0.8}})+(1-0.5e^{-\frac{(x_2-0.8)^2}{0.8}})}{\arctan(100x_1)} \\ \text{subject to } 0\le x_1\le 1 \text{and } 0.0999\le x_2\le 0.1001\text{, or } 0.79\le x_2\le 1. \end{array}} \qquad (10)$$

In Equation (10), parameter $q_1 = q_2$, implies there will not be any spike in the function $T(x, y)$, thus the search space will not be separated into two parts. Indeed, there is only one optimal point for $T(x, y)$ at $x_2 \approx 0.28$. However, as we designed a new constraint for the decision variables in Equation (10), we still can produce similar local-global optimality results shown in Figure 16. In this scenario, the global Pareto front and local Pareto front still exists, except they are created by a strict constraint.



(a)

(b)

Figure 16(a) Decision space and constraint local and global Pareto optimal set (b) Objective space and local and global Pareto fronts of function *F2-2*

Under the same conditions, we run four selected MOEAs and the proposed RDGA, given initial population set 1 and set 2 for comparison. Tables 3 and 4 show the indicator values for set 1 and set 2 correspondingly.

Comparing the indicator values in Tables 3 and 4 with those in Tables 1 and 2, we can see that for the function ***F2-2***, the global Pareto fronts, resulted by adding constraints, are easier to be found by

MOEAs than those resulted from objective functions. This occurrence can be explained as the local optimality represented in Equation (8) having multiple-layer pseudo-global Pareto fronts, each of which contributes a new local Pareto front. In this case, instead of finding the global Pareto front, MOEAs are easily trapped by a local or pseudo-global Pareto front. Nevertheless, the local optimality caused by constraints does not enclose these pseudo-global Pareto fronts. The gap between local and global Pareto fronts is completely blank, which means the resulting non-dominated points are most likely located on either of them, thus simplifying the searching complexity.

Table 3 Final simulation results for function *F2-2* by five MOEAs using initial population set 1

	Number of runs	Stop generation	Final average individual rank value	Final average individual density value	Final average generation distance	Number of runs produce pure global Pareto front	Number of runs produce pure local Pareto front	Number of runs produce partial global Pareto front
MOGA	50	10,000	1.21	3.33	0.32	4	18	28
NSGA-II	50	10,000	1	5.01	0.27	6	15	29
PAES	50	10,000	1	3.96	0.35	5	20	25
RDGA	50	10,000	1.13	5.61	0.22	9	13	28
SPEA II	50	10,000	1.08	5.05	0.24	10	15	25

Table 4 Final simulation results for function *F2-2* by five MOEAs using initial population set 2

	Number of runs	Stop generation	Final average individual rank value	Final average individual density value	Final average generation distance	Number of runs produce pure global Pareto front	Number of runs produce pure local Pareto front	Number of runs produce partial global Pareto front
MOGA	50	10,000	1.04	3.20	0.08	45	0	5
NSGA-II	50	10,000	1	4.61	0.03	48	0	2
PAES	50	10,000	1	3.83	0.08	44	0	6
RDGA	50	10,000	1	4.09	0.02	48	0	2
SPEA II	50	10,000	1	4.52	0.02	49	0	1

For the local optimality created by Equation (10), the smaller the constraint range for $x_{2_{global}}$ ($0.0999 \le x_{2_{global}} \le 0.1001$ in Equation (10)) is, the more difficult for MOEAs to find a real Pareto front will be, because the global Pareto optimal set will be a thinner band when the constraint range is small.

5.3 *F3*—MOP with high-dimensional decision space

$$\boxed{\begin{array}{l} \text{Minimize } f_1(x) \text{ and } f_2(x)\text{, where} \\ \qquad f_1(x) = 1 - e^{-4x_1}\sin^6(6\pi x_1) \\ \qquad f_2(x) = g(x)(1-\frac{f_1(x)}{g(x)})^2 \;\; g(x) = 1 + 4(\sum_{i=2}^{5} x_i/4)^{0.25}, \\ \text{subject to } 0 \le x_i \le 1,\;\; i = 1,\ldots,5. \end{array}} \qquad (11)$$

This test function is proposed in [12] as an MOP with a high-dimensional decision space and local Pareto front in the objective space as shown in Figure 17. Figures 18(b) – (f) show resulting Pareto fronts by five chosen MOEAs for a randomly generated initial population, which is plotted as Figure 18(a) with a true Pareto front. The Box plots for the average values of three indicators over 50 runs are illustrated in Figures 19(a), (b) and (c), respectively. The performance measures of $C(X_i, X_j)$ for the comparison sets between algorithms i and j are shown in Figure 20, where algorithms 1 – 5 represent MOGA, NSGA-II, PAES, RDGA and SPEA II in alphabetical order, respectively.

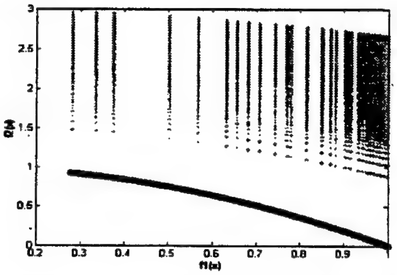
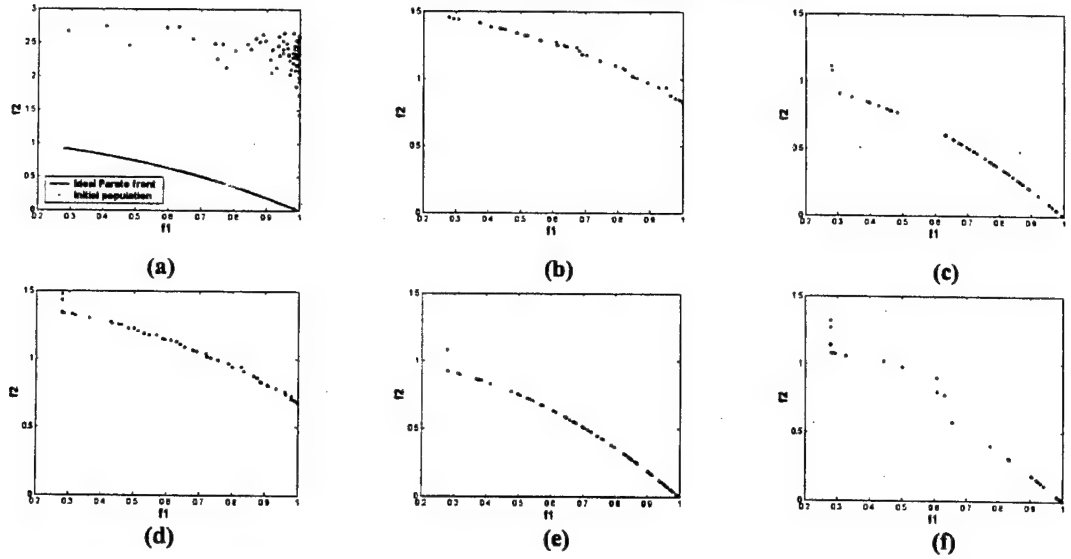


Figure 17 Objective space and Pareto front of *F3*

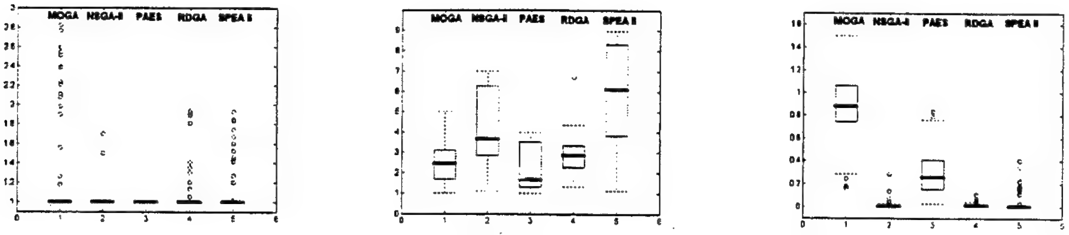


(a) (b) (c)

(d) (e) (f)

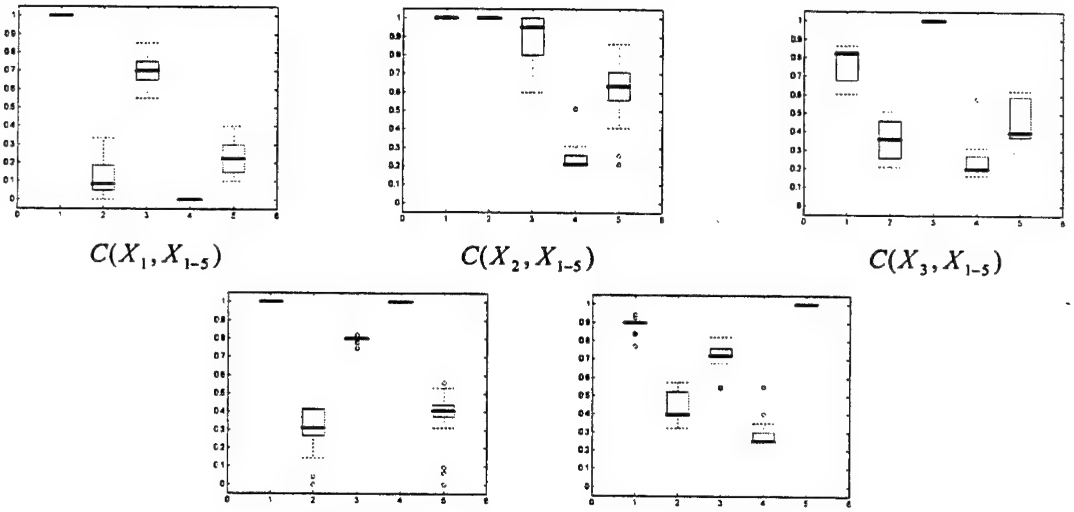
Figure 18(a) Ideal Pareto front and a randomly generated initial population; (b) – (f) Pareto fronts resulted from (b) MOGA; (c) NSGA-II; (d) PAES; (e) RDGA; and (f) SPEA II on test function *F3*

From Figures 18– 20, it is obvious that MOGA has great difficulty in finding the true Pareto front of this MOP. On the other hand, NSGA-II, SPEA II and RDGA can always identify some points on the global Pareto front. Moreover, comparing to NSGA-II and SPEA II, RDGA has the lowest density value, which means RDGA tends to produce a more homogenously distributed Pareto front by minimizing individual's density value independently.



(a) (b) (c)

Figure 19 Box plots of (a) average individual rank value; (b) average individual density value; and (c) average individual distance on test function *F3*



$C(X_1, X_{1\text{-}5})$ $C(X_2, X_{1\text{-}5})$ $C(X_3, X_{1\text{-}5})$

$C(X_4, X_{1\text{-}5})$ $C(X_5, X_{1\text{-}5})$

Figure 20 Box plots based on $\mathbb{C}$ measure on test function *F3*

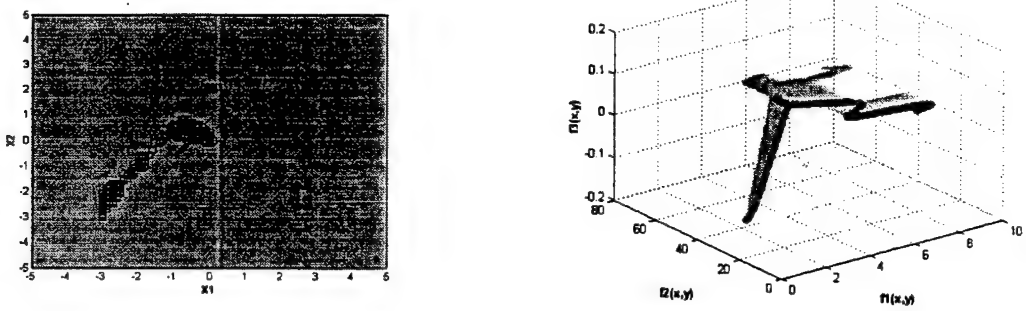
5.4 *F4*—MOP with high-dimensional objective space

Minimize $f_1(x,y)$, $f_2(x,y)$, and $f_3(x,y)$, where

$$f_1(x,y) = 0.5(x^2+y^2)+\sin(x^2+y^2)$$
$$f_2(x,y) = \frac{(3x-2y+4)^2}{8}+\frac{(x-y+1)^2}{27}+15 \qquad (12)$$
$$f_3(x,y) = \frac{1}{(x^2+y^2+1)}-1.1e^{(-x^2-y^2)}$$

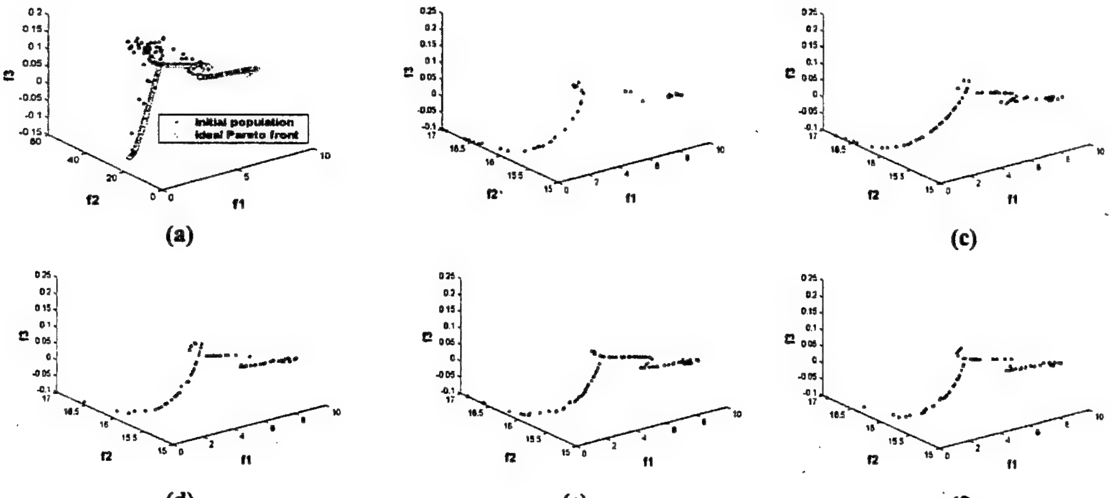
subject to $-30 \le x, y \le 30$.

Originally designed by Viennet [31], this test function has been adopted by many researchers in that it provides three partial-contradict objective functions as shown in Figure 21. Figures 22(b)– (f) show resulting Pareto fronts by five MOEAs for a randomly generated initial population, which is plotted in Figure 22(a) with a true Pareto front. The Box plots for the average values of three indicators over 50 runs are depicted in Figures 23(a), (b) and (c), respectively. The performance measures of $C(X_i, X_j)$ for the comparison sets between algorithms i and j are shown in Figure 24, where algorithms 1– 5 represent MOGA, NSGA-II, PAES, RDGA and SPEA II in alphabetical order, respectively.



(a) (b)

Figure 21(a) Decision space and Pareto optimal set (b) Objective space and Pareto front of function *F4*



(a) (c)

(d) (e) (f)

Figure 22(a) Ideal Pareto front and a randomly generated initial population; (b) – (f) Pareto fronts resulted from (b) MOGA; (c) NSGA-II; (d) PAES; (e) RDGA; and (f) SPEA II on test function *F4*

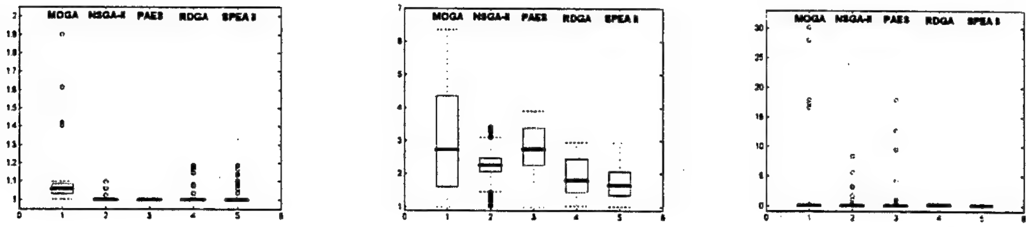


Figure 23 Box plots of (a) average individual rank value; (b) average individual density value; and (c) average individual distance on test function *F4*

Indeed, test function *F4* possesses several challenging characteristics such as a high-dimensional objective space, discontinuous Pareto optimal set and several local minima in objective functions. From the resulting Pareto fronts and Box plots of the performance indicators in Figure 22-23, RDGA, NSGA-II, PAES, and SPEA II all show the ability to approximate the true Pareto front and the population-based

MOEAs (i.e., RDGA, SPEA II and NSGA-II) provide higher *C* values as shown in Figure 24. Furthermore, we can see that RDGA produces the smallest average individual density value and distance value compared with NSGA-II and SPEA II. Because RDGA converts the original objective space into a bi-objective rank-density domain, it is not so sensitive to the complexity of high-dimensional objective spaces. Therefore, RDGA holds the potential promise in solving these types of MOPs.

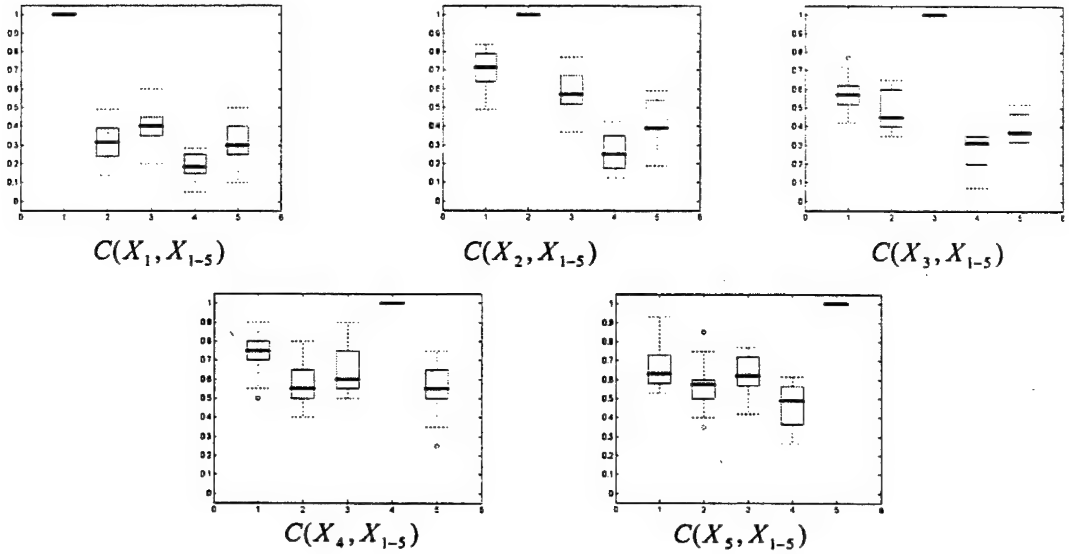
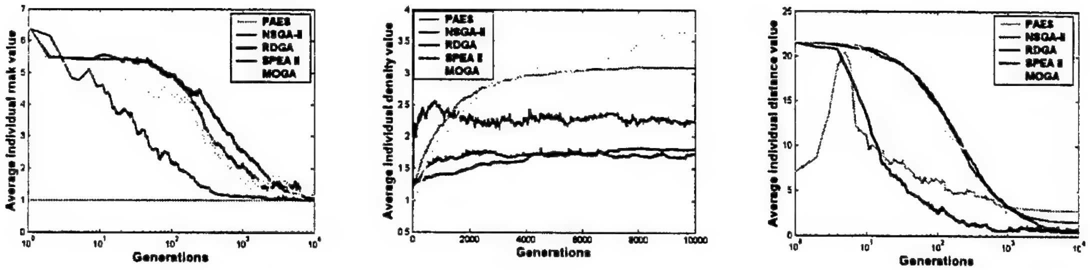


Figure 24 Box plots based on C measure on test function *F4*



(a) (b) (c)

Figure 25 Evolutionary trajectories of (a) rank; (b) density; and (c) distance values by selected MOEAs on test function *F4*

As shown in Figure 25(b) and (c), although NSGA-II performs worse than RDGA and SPEA II in terms of density preservation and distance minimization, it converges relatively fast in the rank domain (Figure 25(a)). This phenomenon can be partially credited to the pure Pareto ranking scheme used by NSGA-II, which will not be affected by the density information during the evolutionary process. However, fast convergence of the rank value does not assure that density and distance values will converge fast as well, and vice versa. As shown in Figures 25(a)-(c), although RDGA converges much slower than the other three population-based MOEAs in terms of rank indicator, it has the fastest convergence speed in terms of a distance indicator compared with all the other selected MOEAs. This effect can be explained by the restricted mating method and "forbidden region" scheme applied by RDGA. On one hand, instead of using the roulette wheel or tournament selection scheme, RDGA randomly selects an individual as one of the parent to mate with the best individual located in the neighboring cells, which ensures the worst individuals have the same probabilities with the elitists to be selected and updated by their better fitted offspring. Although this strategy may sacrifice the convergence speed of an elitist in finding a *single* true non-dominated point, it offers those ill performed individuals a fair chance to catch up with the better ones and draws the entire population to the true Pareto front. On the other hand, the "forbidden region" concept prevents an individual from leading into a wrong direction when the density subpopulation is evolved. In this case, a newly generated offspring can survive not only because it has a lower density value than its corresponding parent, but it also has an equal or higher rank value comparing to the selected parent. For this reason, as an extra constraint of RDGA, the "forbidden region" concept also helps to compress the entire population and push it closer to the true Pareto front. Therefore, both "restricted mating" and "forbidden region" techniques contribute a low variance and fast convergence of the average individual distance value as shown in Figure 23(c) and Figure 25(c). Note that the effect of these two techniques are particularly significant for function ***F4***, which may easily result in an extremely high variance of the distance value during the evolutionary process if an ill-performed individual has never been updated since the beginning. In addition, it is worthy to note that PAES is not a population-based algorithm and only non-dominated individuals are stored in the archive at each generation. These characteristics distinguish PAES from other MOEAs mainly in two aspects: its initial rank and density values are always equal to one, and the average individual rank value will remain to be one during the entire evolutionary process. From the simulation study, although PAES outperforms MOGA for all the test functions, it cannot provide competitive results compared with the other two most advanced MOEAs (i.e., NSGA-II and SPEA II) and the proposed RDGA in terms of rank, density, distance indicators and *C* measure.

6. CONCLUSIONS

In this paper, a new multiobjective evolutionary algorithm—Rank Density based Genetic Algorithm (RDGA) is proposed. RDGA can be characterized as a) simplifying the problem domain by converting high-dimensional multiple objectives into two objectives to minimize the individual rank value and population density value, b) searching for and keeping better-approximated Pareto points by diffusion and elitism schemes, c) incorporating density information into Pareto ranking strategy and d) preventing harmful individuals by introducing a "forbidden region" concept. From the results presented above, RDGA has shown its potential in producing statistically competitive results with the four state-of-the-art MOEAs— Fonseca's MOGA, PAES, NSGA-II and SPEA II on four types of multiobjective optimization problems, which are designed to exploit various complications in finding *near-optimal*, *near-complete* and *uniformly distributed* true Pareto fronts.

For the MOP test functions that only possess discontinuous or concave Pareto fronts, the recent developed approaches—RDGA, NSGA-II, PAES and SPEA-II do not have much trouble in finding some points of the true Pareto front, and RDGA is found to show better performance in keeping the diversity of the individuals along the current trade-off surface, extending the Pareto front to new areas, and finding a well-approximated, non-dominated set. However, without cautious selection of an initial population, an MOP with a feature of local optimality will easily cause most MOEAs problems in finding a pure global Pareto front. A local Pareto front created by constraints may produce less difficulty than what is generated by objective functions if the former one does not contain pseudo-global Pareto fronts. In addition, two complicated MOP test functions with high-dimensional decision space and objective space are examined by RDGA and the selected MOEAs. The experimental results demonstrate that RDGA produces statistically competitive results with other representative Pareto-based MOEAs in finding a *near-optimal*, *near-complete* and *uniformly distributed* Pareto front. Furthermore, as the test functions used in this paper are still far from embodying a complete MOP test suite, a more profound study in developing a general model of MOEA and designing a more representative test function set in this field is absolutely necessary in future work.

REFERENCES

[1] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 257-271, Nov. 1999.

[2] C. L. Hwang and A. S. M. Masud, *Multiple Objective Decision Making—Methods and Applications*, Vol. 164 of *Lecture Notes in Economics and Mathematical Systems*. Berlin, Germany: Springer-Verlag, 1979.

[3] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evol. Comput.*, vol. 3, pp. 1-16, 1995.

[4] S. W. Mahfoud, "Genetic drift in sharing methods," in *Proc. 1st IEEE Cong. Evolutionary Computation*, vol. 1, pp. 67-72, 1994.

[5] A. Petrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proc. 3rd IEEE Cong. Evolutionary Computation*, pp. 798-803, 1996.

[6] S. Ricardo and B. Amnon, "Evolutionary strategies for a parallel multi-objective genetic algorithm," in *Proc. 9th Int. Conf. Genetic Algorithms*, pp. 227-234, 2000.

[7] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 67-82, April 1997.

[8] D. A. Van Veldhuizen and G. B. Lamont, *Multiobjective Evolutionary Algorithm Research: A History and Analysis*, Technical Report TR-98-03, Air Force Institute of Technology, 1998.

[9] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms—part I: A unified formulation," *IEEE Trans. System, Man, and Cybernetics*, vol. 28, pp. 26-37, Jan. 1998.

[10] J. D. Knowles and D. W. Corne, "Approximating the non-dominated front using the Pareto archived evolutionary strategy," *Evol. Comput.*, vol. 8, pp.149-172, 2000.

[11] E. Zitzler, M. Laumanns and L. Thiele, *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*, Technical Report TIK-Report 103, Swiss Federal Institute of Technology, 2001.

[12] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Proc. Conf. Parallel Problem Solving from Nature VI*, pp. 849-858, 2000.

[13] N. Srinivas and K. Deb, "Multi-Objective function optimization using non-dominated sorting genetic algorithms," *Evol. Comput.*, vol. 2, pp. 221-248, 1994.

[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning. Reading*, MA: Addison-Wesley, 1989.

[15] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms*, pp. 93-100, 1985.

[16] M. Valenzuela-Rendón and E. Uresti-Charre, "A non-generational genetic algorithm for multiobjective optimization," in *Proc. 7th Int. Conf. Genetic Algorithms*, pp. 658-665, 1997.

[17] C. C. H. Borges and H. J. C. Barbosa, "A non-generational genetic algorithm for multiobjective optimization, in *Proc. 7th IEEE Cong. Evolutionary Computation,* pp.172-179, 2000.

[18] J. Horn, N. Nafpliotis and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjetcive optimization," in *Proc. 1st IEEE Cong. Evolutionary Computation,* vol. 1, pp. 82-87, 1994.

[19] T. Krink and R. K. Ursem, "Parameter control using agent based patchwork model," in *Proc. 7th IEEE Cong. Evolutionary Computation,* pp. 77-83, 2000.

[20] M. Laumanns, E. Zitzler, and L. Thiele, " A unified model for multi-objective evolutionary algorithms with elitism," in *Proc. 7th IEEE Cong. Evolutionary Computation,* pp. 46-53, 2000.

[21] D. A. Van Veldhuizen, Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. PhD thesis, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.

[22] K. A. De Jong, *An Analysis of Behavior of a Class of Genetic Adaptive Systems*, PhD dissertation, Department of Computer Science, The University of Michigan, Ann Arbor MI, 1975.

[23] Z. Michalewicz, "Genetic algorithms, numerical optimization, and constraints," in *Proc. 6th Int. Conf. Genetic Algorithms*, pp. 151-158, 1995.

[24] C. M. Fonseca and P. J. Fleming, "On the performance assessment and comparison of stochastic multiobjective optimizers," in *Proc. Conf. Parallel Problem Solving from Natrue—PPSN IV,* pp. 584-593, 1996.

[25] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm test suite," in *Proc. ACM Symp. on Applied Computing,* pp. 351-357, 1999.

[26] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, pp. 173-195, 2000.

[27] J. D. Knowles and D. W. Corne, "Benchmark problem generators and results for the multiobjective degree-constraint minimum spanning tree problem," in *Proc. Conf. Genetic and Evolutionary Computation* (*GECCO-2001*), pp. 424-431, 2001.

[28] D. A. Van Veldhuizen and G. B. Lamont, "On measuring multiobjective evolutionary algorithm performance," in *Proc. 7th IEEE Cong. Evolutionary Computation,* pp. 204-211, 2000.

[29] M. Tanaka, "GA-based decision support system for multicriteria optimization," in *Proc. Int. Conf. Systems, Man, and Cybernetics*, pp. 1556-1561, 1995.

[30] K. Deb, "Multiobjective genetic algorithms: problem difficulties and construction of test problems," *Evol. Comput.*, vol. 7, pp. 205-230, 1999.

[31] R. Viennet, "Multicriteria optimization using a genetic algorithm for determining a Pareto front," *International Journal of Systems Science*, vol. 2, pp. 255-260, 1996.

APPENDIX J:

Automatic Frog Calls Monitoring System: A Machine Learning Approach

by

Gary G. Yen and Qiang Fu

International Journal of Computational Intelligence and Applications, **1**(2), 2001, pp. 165-186

International Journal of Computational Intelligence and Applications
Vol. 1, No. 2 (2001) 165–186
© Imperial College Press

AUTOMATIC FROG CALLS MONITORING SYSTEM: A MACHINE LEARNING APPROACH

GARY G. YEN and QIANG FU

Intelligent Systems and Control Laboratory
School of Electrical and Computer Engineering
Oklahoma State University, Stillwater, OK 74078, USA

Received 25 September 2000
Revised 27 February 2001

Automatic recognition of frog vocalization is considered a valuable tool for a variety of biological research and environmental monitoring applications. In this research an automatic monitoring system, which can recognize the vocalizations of four species of frogs and can identify different individuals within the species of interest, is proposed. For the desired monitoring system, species identification is performed first with the proposed filtering and grouping algorithm. Individual identification, which can estimate frog population within the specific species, is performed in the second stage. Digital signal pre-processing, feature extraction, dimensionality reduction, and neural network pattern classification are performed step by step in this stage. Wavelet Packet feature extraction together with two different dimension reduction algorithms are synergistically integrated to produce final feature vectors, which are to be fed into a neural network classifier. The simulation results show the promising future of deploying an array of continuous, on-line environmental monitoring systems based upon nonintrusive analysis of animal calls.

Keywords: Frog calls monitoring, feature extraction, dimensionality reduction, pattern classification.

1. Introduction

Recently, there is an increasing interest and expenditure in environmental monitoring, both in North America and around the world. It is becoming essential to predict and assess the environmental impact of human activities on plants and animals. The populations of certain kinds of animals like birds and frogs are excellent indicators of overall environmental health. As many of the animals in an area may be heard but not seen, it is convenient to rely on their sounds as a means of identification. In many places manual census is not feasible, if not completely impossible. As a result, automatic recognition of animal sounds is considered a valuable tool for biological research and environmental monitoring applications.

The frog is a small and tailless animal. Most frogs have moist skin. They typically live both on land and in water. Toads are very similar to frogs except that toads typically have rough and dry skin and they often live in drier habitats. Frogs and toads are commonly acknowledged as the major divisions in amphibians. Amphibians lead a double life, alternately on land and in water. Typical amphibians include toads, newts, and salamanders as well as frogs. They usually live in temporary or permanent wetland areas. Frogs are of great importance to humans. They are carnivores and consume large quantities of insects, worms, and other small creatures. In turn, they may be a food source for other animals such as snakes. Frogs and toads are integral parts of the food web. Many researchers in different fields are interested in frogs and toads because they are considered to be bioindicators. The health of frog populations is thought to reflect the health of the whole ecosystem. Since early 1980s, scientists have reported startling declines in the populations of some species of frogs.[1] These declines have occurred globally. Although the reason for population decline remains a mystery, there arises a variety of popular hypotheses and possible justifications, such as fluctuations caused partly by climatological changes, increased ultraviolet light due to anthropogenically caused ozone depletion, diseases, and introduction of exotic species (e.g. bullfrogs). All of the above justifications make frog population an excellent indicator of environmental health, particularly in aquatic habitats because of their biphasic (aquatic and terrestrial) life. Causes for frog population declines remain shrouded in mystery despite increased worldwide research efforts. Because of the difficulty and expensiveness of the censusing population of specific frog species, a conclusive analysis based on the estimation of frog population is not yet available. Manual field tracing of frog calls in extremely hot and high humidity wetlands for an extensive period of time is very difficult. In addition, the calling activities of most species are irregular, depending primarily on rainfall and season. As a result, short field trips to these areas are not a reliable method to census the frog populations.

Frogs, as well as birds and whales, have developed the use of sound as the principal means of distant communication. Most species of frogs can produce two types of calls, a distress call and an advertisement call. Both males and females can make distress calls when they are in danger. Only males can produce advertisement calls, which are used to convey such information as location and breeding readiness to both sexes. Advertisement calls can be used to identify the species of frogs. Auditory recognition of frogs is one feasible way to estimate frog population in the area of interest. Therefore, an automatic monitoring system, which remotely monitors calling anurans in the harsh environment, needs to be established with in-place environmental monitoring infrastructure, such as Mesonet.[2]

The monitoring system, which does not require an expert attendance, deploys a directional microphone to capture frog calls continuously in the field, records sound signals into digital audio tapes, and translates them into digital audio data files. Species identification and individual identification within some species can be automatically carried out in this monitoring system. Useful information such as the number of species identified and their approximate population are then transmitted via environmental monitoring network for follow-up decision-making. The successful development of this automatic monitoring system will provide a

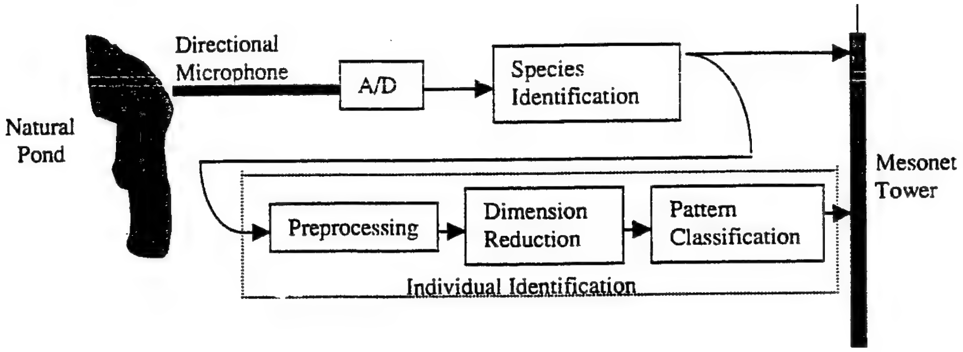


Fig. 1. Automatic frog call monitoring system architecture.

robust measurement to quantify environmental pollution. This system will greatly facilitate research to monitor the amphibian population as an indicator for environmental and water quality. Figure 1 depicts the architecture of the system envisioned.

By developing such an automatic monitoring system which may sustain the harsh environment in the field for a long period of time, it has become possible to *continuously* monitor the frog calls and to reliably estimate the *trend* of frog population within specific species of interest. The remaining of the paper is organized as follows. Section 2 provides a literature review for various representative animal sound identification systems. Section 3 presents the proposed method of species identification, which is proven to be efficient and advanced from those mentioned in Sec. 2. Section 4 shows how individual identification works, which includes three major parts: signal preprocessing, feature vector dimension reduction, and pattern classification. Section 5 discusses the simulation results of species and individual identifications based on the available data sets. Section 6 draws the conclusion of the current research work.

2. Literature Review

Among scarce literature, most of research works on automatic recognition of animal vocalizations focused on species identification, i.e. to identify different species of animals according to recorded vocalizations. Only a few research efforts have been dedicated to quantify the repertoire of a single species and thus estimating population within the same species.[6] Based on different characteristics of various animal sounds, there are different methods to perform automatic or manual recognition. Basically all recognition systems include two stages: digital signal preprocessing and pattern classification. These two stages will be separately discussed as follows.

2.1. *Methods of digital signal pre-processing*

The purpose of digital signal pre-processing is to extract a temporal measurement which contains useful information from the original data. As well known, these large volumes of original data sets generally contain only sparse segments of useful calls. These calls often have weak signal strength and are possibly buried in interference, and usually consist of somewhat similar noises form other animals and the environment. Through the use of pre-processing, we can extract only those useful signals critical for pattern recognition usage.

As with human speech, animal sounds can be sensibly interpreted using a time-frequency representation method. Thus, tools designed for human speech analysis are commonly used for animal sound classification purpose. Generally this includes *time domain methods* such as linear predictive coding,[3,4] *frequency domain methods* such as Fourier transform, *time-frequency domain methods* such as time dependent Fourier transforms, spectrogram,[5,6] and *time-scale domain methods* such as wavelet transforms.[7] In addition, biologists have considered zero-crossing analysis, autocorrelation functions, cepstral analysis, power spectral density (Welch method), and Wigner-Ville transforms as tools for pre-processing of signals. In comparison with the human speech recognition problem, animal sounds are usually simpler to recognize than speech utterances. Their recognition would be an easy problem if it was conducted under similar conditions to that of most successfully deployed speech recognition systems: a single cooperative individual close to the microphone in a quiet environment.[8] Unfortunately, these animal sounds are usually recorded in a much noisy environment which means that we must recognize simpler vocalizations under much more difficult conditions. That is the main difference between human speech recognition and animal sound recognition. Work in the former case focuses on the utterances, while the latter focuses on robustly handling the recording conditions.

The noisy nature of animal sound identification leads to one conclusion: all those signal processing methods mentioned above provide only the *necessary* tools for the process of signal pre-processing. There is still much *sufficient* work to be done to extract the unique features from raw signals.

High-level environmental noise may unnecessarily complicate the identification process than the simulations conducted in laboratory environments. How to realize noise cancellation or noise reduction scheme, especially for those noises with similar frequency range as frog calls, remains a challenging issue.

2.2. *Methods of pattern classification*

Pattern classification forms a fundamental solution to different problems in real world applications. The function of pattern classification is to categorize an unknown pattern into a distinct class based on a suitable similarity measure. Thus, similar patterns are assigned to the same classes while dissimilar patterns are classified into different classes.

Engineers and scientists have developed many methodologies to deal with classification problems. In animal sound identification systems, several methods have been used. The most popular methods are neural network classifier,[3,9] decision-tree classifier,[6] Bayesian Classifier[4] and statistical pattern classifier.[9,10] Some researchers combined more than one method together in their classifier designs. Most existing classifiers can only recognize those already known classes and demands an iterative design process of a long training time. In a near future, the proposed frog call monitoring system is required to be placed in field for real time monitoring. It will not have any prior knowledge about any individual frog call pattern and neither will it know how many individuals are calling during the recording period. In this case, the classifier must be built with on-line learning ability that can learn new patterns in real-time and continuously grow the number of identified individual numbers without re-training. Popular Multi-layer Perceptron network is clearly defeated by this specification. The Incremental Learning Fuzzy Neural Network proposed[18,19] can address this deficiency. It uses an incremental learning algorithm and can detect new classes of signatures and update its parameters while in an operating mode. And it has an on-line (real-time) and fast learning algorithm without knowing *a priori* information.

3. Frog Species Identification

3.1. *Data acquisition of frog call signals*

The proposed frog call monitoring system is based upon in-field acquisition of natural sound signals. One directional microphone, Telinga Pro V Mono Parabolic microphone, was used to collect audio signals. It has frequency response: 40–18,000 Hz: $+/-3$ dB. The microphone was connected with a SONY PCM-M1 digital audio recorder. Audio signals were stored in digital audiotape (DAT). Each DAT has a capacity of 120 minutes. The Turtle Beach System, which includes Turtle Beach Montego II sound card mounted on the PC, Turtle Beach AudioStation, Turtle Beach AudioView and additional supporting software, was utilized to transform signals stored in DAT into the computer in digitized WAVE format. Each WAVE file has PCM (Pulse Code Modulation) format with 8000 Hz sampling frequency, 16-bit accuracy and monotony.

The capacity of each WAVE file is 16 KByte per second. Considering the computational complexity and the characteristics of frog calls, each WAVE file was chosen to be approximately 10-second long. To analyze data in real-time, the computer system should carry on species identification and the corresponding individual identification within 10 seconds. Later a Pentium III 500 PC was used for simulation purpose. It was found that all numerical computations could be done within one or two seconds. That is well below 10 seconds. Hence, a length of 10-second file segment is practically reasonable. After developing the whole hardware system, sound signals can be separated into 10-second long intervals automatically and be analyzed one by one in real-time.

3.2. *Spectrogram analysis*

All frog calls acquisition works were performed in Stillwater, Oklahoma. Frog calls of four different species have been collected from January to May 2000. These were the southern leopard frog (RAUT), American toad (BUAM), streckeris chorus frog (PSST), and spotted chorus frog (PSCL). Advertisement frog calls are species-specific. Some early experiments have shown the fact that a variety of properties of frog calls can be used by one frog species to recognize the vocalizations of their own species.[11] To identify different species of frogs, some useful tools are used to explore properties of different frog calls. The spectrogram, which is frequently used in speech analysis, provides a three-dimensional representation of the sound intensity in different frequency bands over time and thus can be served as a powerful tool for the analysis purpose.

The spectrogram, essentially a type of TDFT, has two general classes: wideband spectrogram and narrowband spectrogram. A wideband spectrogram representation results from a window that is relatively short in time. It has poor resolution in frequency-domain and good resolution in time-domain. While a narrowband spectrogram uses a longer window to provide higher frequency resolution and with a corresponding decrease in time resolution.[12] In this research, single pitches of each frog call need to be separated for identification and analysis purposes. These single pitches usually have very short durations (10–50 ms). As a result, wideband spectrogram which has relatively good resolution in time-domain is being used here. Figure 2 shows typical spectrograms of the four species being studied. All unique call patterns are within the rectangle areas. Notice that different noises occupy nearly all frequency bands. Obviously, each species has unique properties to distinguish them from each other. These include the call rate, the call duration, the amplitude-time envelope, the waveform periodicity, the pulse-repetition rate, the frequency modulation, the frequency, and spectral patterns.[11] Since all frog call signals occupy frequency bands well below 4 KHz, according to sampling theory, the sampling rate (*Fs*) of 8 KHz is a reasonable choice. These four species calls are all within certain frequency bands. According to call patterns, they can be divided into three types. Type I includes RAUT and PSCL. Usually, one frog of these types makes one to several calls at one time. Each call is composed of several pitches (pulses) with similar repetition rate. PSST can be seen as Type II. Each call of PSST is only composed of one single pitch. Type II can be considered as the simpler case of Type I. Type III contains BUAM. This call usually lasts for minutes, which can be regarded as continuous calls. Each call of this type is composed of many single pitches with similar repetition rate. Table 1 describes some typical features of these four different calls.

Generally speaking, one single pitch is the basic element of all different frog calls. Though each pitch may have one peak frequency value, basically the sound energy is well distributed within the whole frequency band. Also, different individuals within the same species may have slightly different peak frequency value. According to different call patterns shown in Fig. 2, two general methods can be used to carry

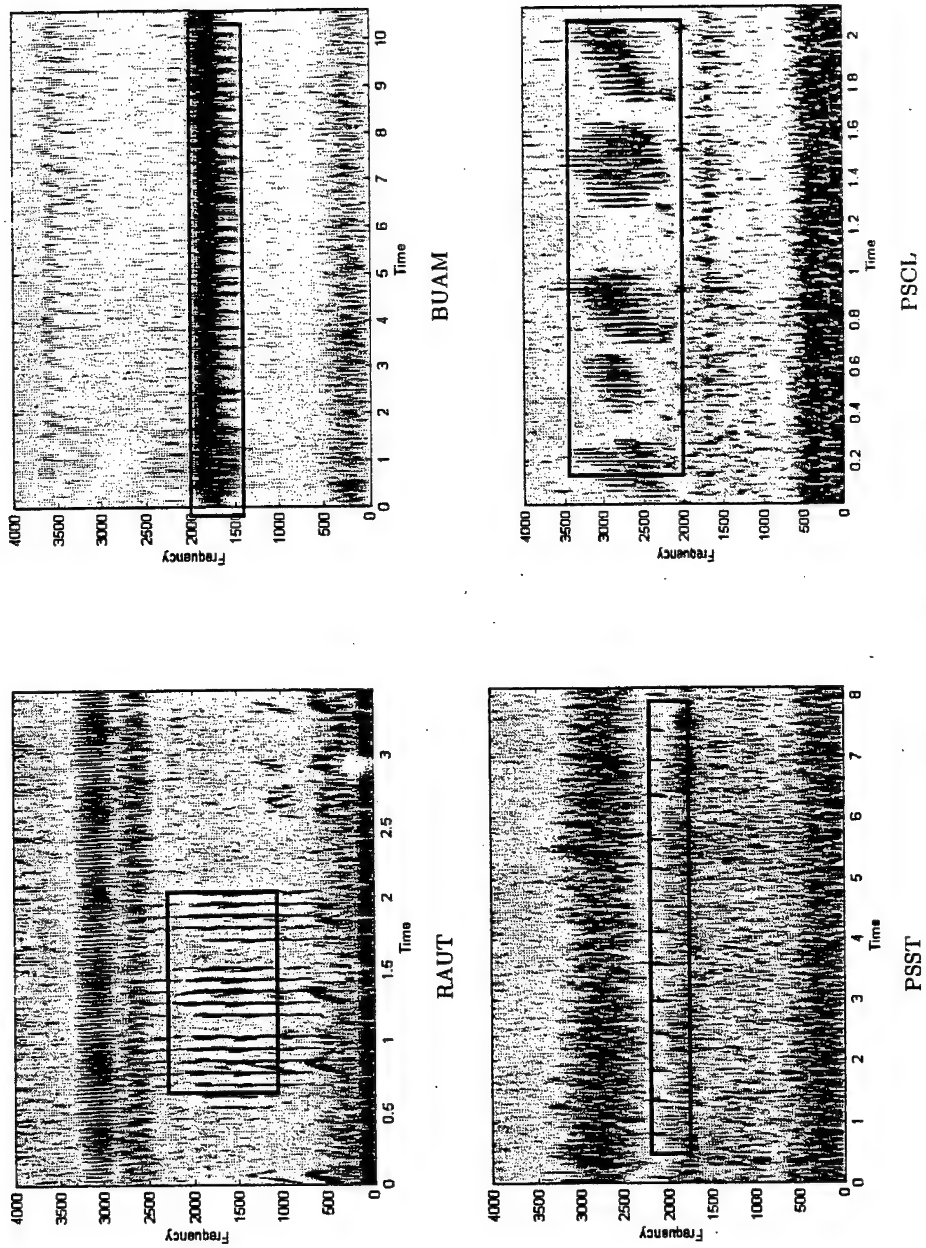


Fig. 2. Spectrograms of four different species.

Table 1. Call patterns of different species.

Species	Main Frequency Band (Hz)	Number of Pitches Within One Call	Pitches Repetition Rate Within One Call (s)	Duration of One Single Pitch (s)
RAUT	800–2200	3 ~ 9	0.07 ~ 0.09	0.02 ~ 0.03
BUAM	1600–2000	Hundreds	0.04 ~ 0.05	0.03 ~ 0.04
PSST	2000–2500	1	N/A	0.04 ~ 0.05
PSCL	2300–3100	7 ~ 14	0.02 ~ 0.03	0.01 ~ 0.015

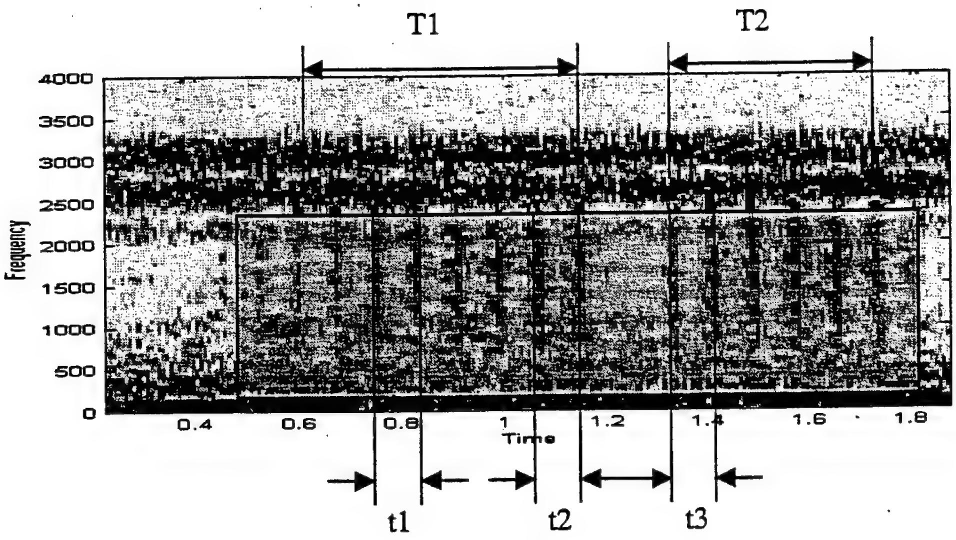


Fig. 3. Spectrogram of RAUT call.

out species identification. One for Types I and II and the other for Type III. Since RAUT and PSCL are similar and PSST is simpler than these two, methods proposed for detecting RAUT will be explained in detail as an example and methods for identifying BUAM will also be illustrated.

3.3. *The proposed filtering and grouping algorithm*

Before any identification work can be carried out, characteristics of frog calls must be carefully studied. Then certain suitable features may be extracted to implement the species identification. Figure 3 showed some details about the spectrogram of two RAUT calls. By looking at this figure, some useful information may be identified. This spectrogram uses a window of length 128 and a FFT of length 256. Two calls made by the same frog are shown in the shaded area. The frequency ranges from 800 Hz to 2400 Hz. Those outside the shaded area are all different background noises. In this case, the noise level is high. Each call is composed of several pitches, which can be seen as vertical striations in the shaded area. The lengths of these two calls are T1 and T2, respectively. The pitch repetition rates within one call are roughly the same. That is, $t1 \approx t2 \approx t3$. After carefully studying all available RAUT call signals, we can make the following observations: (1) All RAUT calls have most energy within 1000–2000 Hz; (2) All RAUT calls are composed of several pitches; (3) The pitch repetition rate within one call is approximately 0.07–0.09 second; and (4) Single pitch length usually ranges from 0.02–0.03 second.

Based on these heuristics, we can use one simple but effective algorithm to identify RAUT calls from the whole data set. We choose one period (roughly 3.5 second) of original sound signals $Y(t)$ as an example. Since RAUT calls are within a certain frequency band, one IIR bandpass filter (BPF) is used to filter other irrelevant noises outside the specified frequency range. Since Chebyshev Type II filters are monotonic in the passband and equiripple in the stopband, they can largely sustain the magnitude of signal components within the passband and attenuate unwanted signal components to the same level, no matter if they are in high frequency band or low frequency band. For these justifications, the Chebyshev Type II filter was chosen to design the BPF. The desired passband is set from 1000 Hz to 2000 Hz. Listed below are parameters designed specially for this BPF[13]:

(1) Passband corner frequency W_p: [1050 Hz 1950 Hz]/(Fs/2)= [0.2625 0.4875].
(2) Stopband corner frequency W_s: [950 Hz 2050 Hz]/(Fs/2)= [0.2375 0.5125].
(3) Passband ripple R_p (the maximum permissible passband loss in decibels): 1 dB.
(4) Stopband attenuation R_s (the number of decibels the stopband is down from the passband): 50 dB.

Based on these parameters, the lowest possible order of Chebyshev Type II filter is 12 and the Chebyshev Type II cutoff frequency, W_n, that allows it to achieve the given specifications is [0.2472 0.5089]. The filtered signal after this BPF is denoted by $Y_{\mathrm{bp}}(t)$. Then $Y_{\mathrm{bp}}(t)$ is squared to get $Y_{\mathrm{sqr}}(t)$. Using the threshold those small values of $Y_{\mathrm{sqr}}(t)$ are zeroed out. The signal denoted by $Y_{\mathrm{thres}}(t)$ represents the signals after thresholding. The threshold, M, is set to be one positive number multiplying the mean value of signals $Y_{\mathrm{sqr}}(t)$, which is proportional to the *average power* of $Y_{\mathrm{bp}}(t)$. In this case, M is set to be $10 \times \mathrm{mean}(Y_{\mathrm{sqr}}(t))$. In $Y_{\mathrm{thres}}(t)$, signals below this threshold (small spikes) are set to zero value so as to further reduce possible noise level.

Single pitches of RAUT calls usually lasts 0.02 second to 0.03 second long and the pitch repetition rate within one call ranges from 0.07 second to 0.09 second. The grouping algorithm is trying to identify those spikes that tend to belong to one single call and group them together. Thus, the time intervals of one single call can be detected. First, by gathering those signals very closely to each other together, the time intervals of those possible isolated spikes (pitches) are detected. Then by checking the duration of each short time interval, those with too short or too long of intervals that obviously are not single pitches are thrown away. The second step is to group those pitches that tend to belong to one call together by

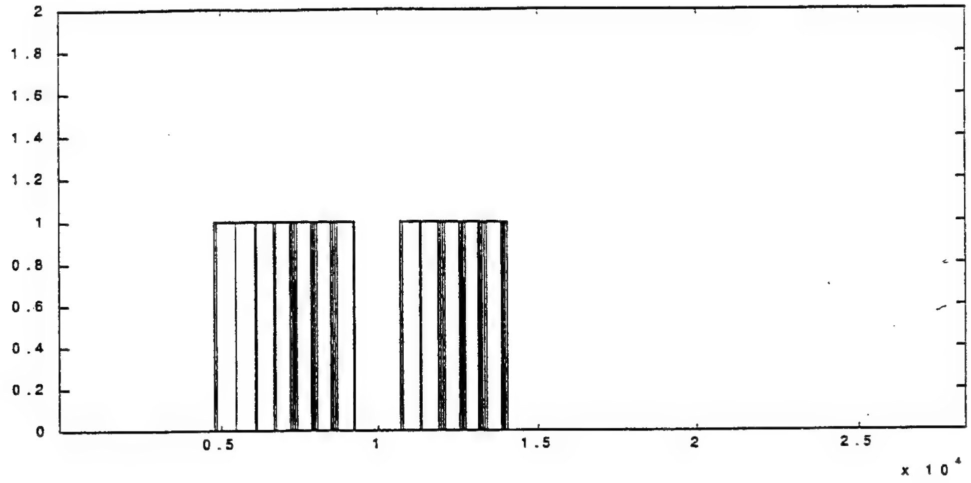


Fig. 4. Two intervals detected by grouping algorithm.

checking the intervals between adjacent pitches. Thus, possible time intervals of single frog calls are acquired. Pitches too far away cannot be clustered together since they tend to belong to different calls. Finally the lengths of possible single calls are checked. Since one RAUT call is often composed of at least three strong isolated pitches, those intervals, which are not long enough, are discarded. After these three steps, real RAUT call intervals have finally been detected and isolated. Figure 4 shows the result of two calls along with several pitches separated by the proposed grouping algorithm. The first call contains eight pitches. It starts from point $t = 4861$ and ends at point $t = 9261$. So the duration of this call is 0.55 second $((9261 - 4861)/8000 = 0.55)$. The second one contains six pitches. It starts from $t = 10\,686$ and ends at $t = 14\,013$. The duration of this call is 0.416 second. The intervals of these single pitches have also been stored for possible further analysis such as individual identification. In Fig. 4, the darker the shaded area, the stronger the single pitch is.

Compared to other species identification methods proposed for marine mammal like whales and dolphins,[14–16] we propose herein a simpler algorithm, which requires much less computation. For the incoming N-point data set, the computational complexity of the proposed filtering and grouping algorithm is at $O(PN)$. While all methods listed in literature require further processing of data with additional pattern classification. The method proposed here does not need this step at all. As mentioned before, calls of Type II (PSST) are the simpler form of those of Type I. So just by changing some parameters, this filtering and grouping algorithm may also be used to identify Type II frog calls.

Since many species of frogs tend to make sounds in a similar pitch/pulse repetition mode, it is convenient to use this filtering algorithm to identify these species.

In doing so, parameters such as passband of BPF and some criteria about pitch duration and call length need to be changed accordingly. There is no need to use the BPF and grouping algorithm to identify species with continuous calls like BUAM. Fast Fourier transform (FFT) is enough to accomplish this task. The main frequency band of BUAM calls is [1600 Hz 2000 Hz]. Usually, peak frequencies of BUAM calls are within a range of [1650 Hz 1850 Hz]. The pitch repetition rate is roughly $0.04 \sim 0.05$ second, that is $320 \sim 400$ points and the duration of one single pitch is usually $0.03 \sim 0.04$ second, that is $240 \sim 320$ points. So a 512-point time interval at anytime within a call is sufficient to cover most parts of one single pitch. Specifically, a 512-point data set was extracted every two seconds along the data file. For one data file approximately 10-second long, a total of four or five data sets can be extracted. FFT was performed on each data set. Then peak frequency values within [1000 Hz 2000 Hz] of each result were checked to see whether they are still within [1650 Hz 1850 Hz]. If so, there must be BUAM calling in this period. Ideally, this method is reliable at detecting continuous calls that have peak frequency values within a narrow frequency band. But if there is some kind of continuous noise, which happens to have the peak frequency value in the same frequency range, a mismatch is unavoidable.

4. Frog Individual Identification

4.1. *Individual identification overview*

Individual identification refers to identifying individual frogs within the same species and estimating the number of the identified frogs. Not all species of frogs are available for this task. The underlying assumption of automatic individual identification is that human experts may distinguish different individuals within the same species. If so, their knowledge can be used for analysis purposes and make the automatic identification possible. If human experts cannot tell the difference of one call from another, it is unlikely for a machine to tell the difference without a guideline. For example, one BUAM frog usually makes calls continuously for several tens of seconds or even minutes. There may or may not be other individual calls at the same time. Even an expert cannot tell the difference. In this case without any specific sample or prior knowledge, no baseline can be established. Within the four species of interest, only RAUT can be identified by human ear individually, so our research will focus on individual identification within this species only.

Before individual identification can be examined, typical samples of individual calls must be collected first. In the species identification stage, individual RAUT calls with several pitches have been extracted as seen in Fig. 4. Since one pitch is the basic element of each RAUT call, we may choose one typical pitch as the sample for one RAUT call. The duration of one single pitch ranges from 0.02 second to 0.03 second, that is $160 \sim 240$-point data segment. A finite duration window $w[n]$ is applied to original signal $Y[n]$ prior to any signal analysis. This produces the finite length sequence $v[n] = w[n]Y[n]$. There are many kinds of windows in digital signal processing such as Bartlett, Hamming, Hanning, Blackman, Kaiser, and rectangular window. Here, we choose one non-overlapping 512-point Hamming window. The center point of the Hamming window was chosen to be at the maximum filtered value ($Y_{\text{bp}}(t)$) within one call. Thus, the strongest pitch within one call has been extracted and serves as the sample vector for this frog call. For each RAUT call, we can extract one 512-point data segment as its sample vector. All future works are based on analysis of this data segment.

4.2. *Feature extraction algorithm*

Feature extraction involves preliminary processing of signals to obtain suitable parameters that reveal distinguishable nature of the specific kind of signal. The aim of feature extraction is to devise a transformation that extracts the signal features hidden in the original domain. Corresponding to different characteristics of signals, different transformations should be properly selected to extract those most typical features from the original signal domain, thus to make the following step of signal analysis, which, in this case, is the pattern classification, much easier.

Many signals require a more flexible approach — one where we can vary the window size to determine features of these signals more accurately either in time or in frequency. The Wavelet analysis employs a windowing technique with variable-sized regions. Wavelet analysis allows the use of long time intervals where we want more precise low frequency information and short regions where we want high frequency information. Wavelet analysis does not use a time-frequency region, but rather a time-scale domain. It is capable of revealing aspects of data that other signal analysis techniques miss, aspects such as trends, breakdown points, and discontinuities in higher derivatives, and self-similarity. Unlike the Wavelet Transform, which decomposes only low frequency components of a signal; Wavelet Packet Transform (WPT) decomposes both the low frequency and high frequency components. This rich abundant information with arbitrary time-frequency resolution can allow extraction of features that combines both stationary and non-stationary characteristics. However, one deficiency that wavelet bases inherently possess is the lack of translation invariant property. A signal with a time shift does not result in the time shifted wavelet packet coefficients. Direct assessment from all wavelet packet coefficients often turns out to be tedious or leads to inaccurate results. Here, we introduce the idea of wavelet packet node energy.[17] Each cell $w_{i,j}$ refers to one node in the decomposition tree, here i is the scaling parameter and j is the oscillation parameter. We call each (i, j) as a wavelet packet node. Each (i, j) has K coefficients, $w_{i,j,k}$. For one signal of length 2^N, the maximum value of i is N, for each i, the maximum value of j is $2^i - 1$ and the maximum value of k is 2^{N-i}.

The wavelet packet node energy is defined as:

$$e_{i,j} = \sum_{k=1}^{K} w_{i,j,k}^2 \tag{4.1}$$

which measures the signal energy contained in some specific frequency band indexed by parameters i and j. In our case, each wavelet packet node energy value was defined as an individual feature component and was used as a robust rudimentary exploration of the specific signal features. For an r level decomposition, we can get a total of $2^1 + 2^2 + \cdots + 2^r = 2^{r+1} - 2$ sets of node energy coefficients. These coefficients are final results extracted from each frog call signal by using wavelet method.

4.3. *Dimension reduction/feature selection algorithm*

Notice using WPT may produce a high dimensional feature vector for each individual frog call signal. Direct manipulation on a whole data set is not feasible because of high dimensionality of data and the existence of undesired components that make the classification unnecessarily complicated. Thus, it is desirable to use lower dimensional feature vectors as input for the pattern classifier. To reduce dimension of feature vectors, one idea is to find a linear transformation that maps high dimensional data onto lower dimensional space, the other is to select those feature components that contain most discriminant information and discard those that provide little information which is useful for classification purpose. Here, the second method is chosen for the dimension reduction purpose. Specifically, the feature component $\{f_k | k = 1, 2 \ldots, n\}$ is ranked by

$$J(f_1) \geq J(f_2) \geq \cdots \geq J(f_d) \geq \cdots \geq J(f_n) \tag{4.2}$$

where $J(\cdot)$ is a criterion function for measuring the discriminant power of a specific feature component, f_k. Feature subsets can be chosen from those features having larger criterion function values.

In this study, a simple while efficient criterion function known as Fisher's criterion[17] was adopted. For a two classes problem, it is given by

$$J_{f_k}(i,j) = \frac{|\mu_{i,f_k} - \mu_{j,f_k}|^2}{\delta^2_{i,f_k} + \delta^2_{j,f_k}} \tag{4.3}$$

where μ_{i,f_k} and μ_{j,f_k} are the mean values of the kth feature, f_k, for class i and j; δ^2_{i,f_k} and δ^2_{j,f_k} are the variance of the kth feature, f_k, for class i and j correspondingly. For multiple classes (class number equals to L) case, the general approach is to take summation of the pairwise combinations of $J_{f_k}(i,j)$:

$$J_{f_k} = \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} J_{f_k}(i,j) \tag{4.4}$$

as an estimation of discriminant power for the specific feature f_k. Equation (3) provides a measure to evaluate the effectiveness of the "global" feature that is simultaneously suitable to differentiate all classes of signals. For small classes case, this approach may be sufficient. When the number of classes increases, this equation becomes more ambiguous. A large value of J_{f_k} may be due to the accumulations of many relatively small values (an unfavorable case) or to a few significant terms with negligible majority (a favorable case). Also a feature with large $J_{f_k}(i,j)$ value to class i and j may have very small discrimination power for other classes, thus this makes J_{f_k} also very small. To avoid these problems, two methods are proposed as possible alternatives.

Method I. Instead of trying to select features, which are effective for the entire multi-class problem globally as measured by Eq. (4), a feature subset based on Eq. (3) was selected for each possible pair of classes. Then the union of feature components selected from each pair of classes was taken to form the final feature vector. Specifically, given a L-class problem with n feature components, the process is detailed in the following steps:

(1) For each possible class pair $\{(i,j)|i=1,2,\ldots,L-1,\ j=i+1,\ i+2,\ldots,L\}$, calculate the discriminant power measure for each feature component, f_k, using Eq. (4.3).

(2) For each class pair, sort $J_{f_k}(i,j)$ such that:

$$J_{f_1}(i,j) \geq J_{f_2}(i,j) \geq \cdots \geq J_{f_d}(i,j) \geq \cdots \geq J_{f_n}(i,j)\,. \tag{4.5}$$

Determine the feature subset $F_{i,j}$ for each class pair by selecting d feature components that have maximum $J_{f_k}(i,j)$ value:

$$F_{i,j} = \{f_k|k=1,2,\ldots,d\}, \quad i=1,2,\ldots,L-1; \quad j=i+1,i+2,\ldots,L\,. \tag{4.6}$$

3. Form the final feature set by taking the union of each feature subset.

$$F_{\text{final}} = \bigcup_{i=1}^{L-1}\bigcup_{j=i+1}^{L} F_{i,j} \tag{4.7}$$

Method II. This method is based on a similar idea of Method I. Compared to Method I, Method II may choose a different number of feature components from each class pair, thus more reasonable feature components are expected with this method. The first step is the same as that of Method I. In the second step, after $J_{f_k}(i,j)$ were sorted in descending order, the whole data set was normalized. That is:

$$J = \sum_{k=1}^{n} J_{f_k}(i,j) \tag{4.8}$$

$$J'_{f_k}(i,j) = \frac{J_{f_k}(i,j)}{J} \quad k=1,2,\ldots,n\,. \tag{4.9}$$

Set one threshold value $H \in (0,1]$. Determine the feature subset $F_{i,j}$ for each class pair by selecting D feature components that have maximum $J_{f_k}(i,j)$ value. D must satisfy:

$$\begin{cases} \sum_{k=1}^{D} J'_{f_k}(i,j) \geq H \\ \sum_{k=1}^{D-1} J'_{f_k}(i,j) < H \end{cases} \tag{4.10}$$

$$F_{i,j} = \{f_k | k = 1, 2, \ldots, D\}, \quad i = 1, 2, \ldots, L-1; \quad j = i+1, \quad i+2, \ldots, L\,. \tag{4.11}$$

The third step is the same as that of Method I. By using Method II, different numbers of feature components may be extracted from each class pair. If two classes are well separable, there must exist a few feature components that contain much larger $J_{f_k}(i,j)$ values than most other feature components. If two classes are not well separable, then most feature components tend to have similar relatively small $J_{f_k}(i,j)$ values. By setting this threshold, we may choose fewer feature components from well separable class pairs and more feature components from those classes that were difficult to separate. Thus, we may get feature subsets with relatively low dimension while still containing high discriminant power.

4.4. *Pattern classification algorithm*

Once after suitable feature components have been extracted from the original feature set, it is then necessary to determine individual frogs based upon these features. Neural network and a variety of multivariate statistical methods have been used for pattern classification problems. Neural network classifiers are widely used because they are universal function approximators and because of their nonlinear nature, they have the ability to capture the underlying non-linearity from the incoming data. However, for Multi-layer Perceptron (MLP) network, existing patterns must be used to train the network and the classifier can only detect those already existed classes. That is, in our case, *a priori* knowledge for each individuals of RAUT in one area is given. If one new RAUT frog makes a call and its call features are fed into the classifier, the MLP classifier may not identify it as a new one and possibly will classify it into one already existing RAUT frog class. As a result, the classifier must be built in with on-line learning ability that can learn new patterns in real-time and grow continuously with the number of identified individual numbers without re-training. MLP is clearly defeated by this specification. The Incremental Learning Fuzzy Neural Network (ILFN) developed in Refs. 18 and 19 can address this deficiency. It uses an incremental learning algorithm and can detect new classes of patterns and update its parameters while in an operating mode. And it has an on-line (real-time) and fast learning algorithm without knowing *a priori* information.

5. Test Results

In this section, natural frog calls collected via Telinga Pro V Mono Parabolic microphone mounted at various lakesides in Stillwater, Oklahoma were used to validate the feasibility of the proposed species and individual identification methods.

5.1. *Results for species identification*

For species identification, one DAT with a total length of 50 minutes was chosen as the sample. It contains frog calls of all four species obtained from several lakesides nearby. Each species contains several different individuals. The entire DAT data were manually saved into the PC in WAVE format. Each file segment was approximately 10 seconds long. Each data set was fed into four different programs that identify one species correspondingly. The goal is that each program may be able to identify all clear calls of that species and does not miscount other signals (e.g. calls made by other species). It is much more desirable for our system to fail to recognize a call (a false negative) than to incorrectly indicate the call of a particular species is present (a false positive). It is crucial then to choose parameters such that false positives are minimal. For example, to identify RAUT and PSCL frogs, according to call properties generalized in Sec. 3, we narrow down the ranges of pitch duration thus to avoid mismatch with other short duration spikes. Also, in the clustering algorithm, to choose possible call signals and discard the false impulses we do thresholding on the squared signal. If the threshold value is too big, more irrelevant signals will be discarded but some portion of true frog call signals (those pitches in the beginning or in the end of one call) will also be thrown away due to little energy they have. There exists a trade-off between recognizing more false negative and less false positive. In practice, by carefully adjusting various parameter values, the result for species identification is quite promising. Within this sample period, there are hundreds of calls belonging to four different species. Except for some weak calls and some calls obscured by environmental noise, most clear calls can be detected and identified as belonging to correct class with nearly 100% accuracy. For frog species of BUAM and PSST, the results are found to be perfect.

Yet, there do exist a few mismatches when identifying species RAUT and PSCL. The noise causes part of the problem. In most natural situations, background noise is extremely high and its temporal and spectral structure are complex and variable. In this DAT tape, there always exists three types of noise:

(1) Noises made by other living creatures: including calls made by other frog species and some insects like crickets. Occasionally there exists some dog barks and human speech if the pond is close to human community.
(2) Noises made by natural phenomena: including wind noise and rain noise.
(3) Noises made by vehicles: including noises made by automobiles.

Among these three types of noises, 1 and 2 occur more frequently. If the frequency band of these noises is different from that of the specific frog species, they can be removed in the stage of filtering. Or if the spectrogram of these noises did not appear to be a steady pulse repetition mode just like those of RAUT and PSCL, they can also be eliminated in the stage of clustering. But if the main frequency band and pulse shape of that kind of noise are quite similar to those of frog species, a mismatch is inevitable by using the proposed identification method. For the third type of noise, although it happens occasionally, if the noise level is high, frog calls may be occluded and sometimes mismatch may also occur.

5.2. *Results for individual identification*

5.2.1. *Data segmentation*

Individual identification is based on the results of species identification. After one call of RAUT species has been identified, its calling period has also been determined simultaneously. Then a non-overlapping 512-point Hamming window was used to extract a 512-point time series data segment from one RAUT call as its sample vector. The length of window guarantees to contain at least one pitch (the strongest one) within this sample vector. For the same species, it is reasonable to assume that call patterns of different individuals can be fully explored by analyzing one single pitch.

Before this data segment can be used for further analysis, the mean value of this segment is calculated first and subtracted from the whole data set. This is because signals with non-zero mean may produce incorrect spectrum estimates especially in low frequency band. Subtracting mean value from the signal often leads to a better estimate at neighboring frequencies.

5.2.2. *Generation of training/testing data set*

Because frogs are sensitive to sudden changes of environment, their calls are difficult to collect. Also human experts usually have limited capability to identify different individuals. For these reasons there are a total of 66 data sets that have been identified, which correspond to four different individual RAUT frogs. For these 66 data sets, each time 44 data sets were randomly chosen as training data while the remaining 22 data sets were used for testing. The distribution of training and testing data sets are shown in Table 2.

5.2.3. *System description*

After 66 of 512-point feature vectors are extracted, Wavelet Packet Transform (WPT) is used for feature extraction. In addition, Linear Predictive Coding (LPC), and Time-Dependent Fourier Transform (TDFT) are used for comparison. For TDFT and WPT, two different dimension reduction algorithms are used to derive the final feature vector, which will then be fed into a neural network classifier. For fairness of the comparison, MLP is used here as a neural classifier. It was

Table 2. Number of individual samples and the distribution of training/testing sets.

RAUT Individual	Total Number of Data Sets	# of Data Sets for Training	# of Data Sets for Testing
Frog 1	16	11	5
Frog 2	20	13	7
Frog 3	17	11	6
Frog 4	13	9	4

found in Ref. 18 that ILFN will outperform the MLP in all classification tasks. The steps summarized below provided the details for each feature extraction algorithm considered.

LPC:

(1) Determine number of LPC coefficient, p, according to mean square error (MSE) between filtered values and actual values. It was found at $p = 12$ that the MSE is acceptable.
(2) For each sample vector, determine 16 time domain LPC filter coefficients.
(3) Calculate FFT of 16-point LPC coefficients and obtain nine unique spectral magnitudes.
(4) Normalize these nine spectral magnitudes to derive final feature vector.

TDFT:

(1) Calculate 512-point FFT for each windowed data set and obtain 257-point spectral magnitude vector.
(2) Use feature reduction Methods I and II, set parameters d (defined in Method I) and H (defined in Method II) and derive the corresponding feature subsets.
(3) Normalize these two feature subsets and acquire the final feature vector.

WPT:

(1) Perform eight-level wavelet packet decomposition for each 512-point data set by using Daubechies eight-point wavelet function.
(2) Calculate wavelet packet node energy according to Eq. (4.1) and obtain the 510-point feature vector.
(3) Use feature reduction Methods I and II to derive the corresponding feature subsets.
(4) Normalize these two feature subsets and obtain the final feature vector.

5.2.4. *Test results*

After training data were obtained by three feature extraction algorithms, LPC, TDFT and WPT, as well as two feature reduction algorithms, Methods I and II, they were fed into a MLP neural classifier. By checking the final results of the classifier to those testing data sets, some conclusions may be drawn pertaining to which method is better and which one is not. For dimension reduction Methods I and II, parameter d and H should be carefully chosen so that reduced feature vectors with same or similar dimension may be generated. There are a total of 66 data sets available. In each test, 44 of them were randomly chosen as training sets while the remaining 22 were used as testing sets. This process is repeated 1000 times. The number of training sets and testing sets within one class (calls produced by the same frog) are fixed, as seen in Table 2. The mean value of these 1000 simulations was calculated as indicator for test performance. The variance is also computed. If all variances are not high and in the same level, then the performance of the whole system can be regarded as stable. The network architectures are N-N-4 (with N neurons in the only hidden layer) and N-10-10-4 (with 10 neurons in the first and the second hidden layers), where N is the dimension of the final feature vector. In the learning phase, the network is trained until the mean square error is below 0.001, or the maximum epochs (set to 1000) is reached. The resilient backpropagation algorithm[20] is used to train the network. In training we can make the desired output of MLP to be a perfect decision, i.e. one 1 and three 0 second. But the classifier will not produce such perfect decision in the testing process. Usually each output will be between 0 and 1. Here, we use the maximum output value as the most likely individual frog. In all cases, a clear winner can always be identified. The classification results are shown in Tables 3–5. Mean is referred to as mean accuracy. It has a range from 0 to 1. Var. is referred to as variance of the 1000 runs. The training accuracy is always 100% in all cases and the corresponding variance is 0. So these tables only show test results for different methods.

First, examine the results of LPC method. The classifier can only correctly classify roughly half of the test samples, which is not good and much lower than the results of TDFT and WPT methods. It can be noticed that these samples contain a large amount of noise. A rough estimate to some data files shows an average SNR (signal to noise ratio) of −3 dB. The noise significantly deteriorates the performance of LPC filter and finally leads to poor performance of the neural network classifier.

On the average, neural network classifiers based on WPT method acquires the accuracy of classification 8% higher than those based on TDFT method. A Wavelet based method provides a better time-frequency resolution and they are more efficient than Fourier based methods for non-stationary signal analysis. To compare the performance of two feature reduction methods I and II, some conditions have

Table 3. Test Results for LPC method.

LPC		N-N-4
N = 9	Mean	0.5068
	Var.	0.0104

Table 4. Test results for TDFT method.

TDFT	N		N-N-4	N-10-10-4
Method I	17	Mean	0.6082	0.6089
$d = 4$		Var.	0.0075	0.0099
Method II	18	Mean	0.6218	0.6188
$H = 0.1$		Var.	0.0115	0.0089
Method I	33	Mean	0.6471	0.6505
$d = 8$		Var.	0.0093	0.0098
Method II	33	Mean	0.6330	0.6377
$H = 0.16$		Var.	0.0094	0.0093

Table 5. Test results for WPT method.

WPT	N		N-N-4	N-10-10-4
Method I	19	Mean	0.6827	0.6809
$d = 5$		Var.	0.0097	0.0105
Method II	18	Mean	0.7118	0.7164
$H = 0.06$		Var.	0.0068	0.0100
Method I	31	Mean	0.6609	0.6891
$d = 8$		Var.	0.0095	0.0102
Method II	33	Mean	0.7218	0.6955
$H = 0.1$		Var.	0.0076	0.0150

been set. Basically, when the dimension of final feature vectors is the same or quite similar, using Method II may extract feature components with more discriminant power thus to make the performance of neural network classifier better. This is especially true in WPT case, in which by using Method II the accuracy of the classifier is on the average 3% higher than that of using Method I. This also substantiates our assumption that we may choose fewer feature components to distinguish those easily separable classes and choose more feature components to distinguish those relatively not so easily separable classes. By this way, more feature components that contain the most discriminant power may be included with limited feature vector dimension. It is observed that WPT method exerts a large amount of computation load compared to TDFT method. If WPT method is to be used, it is preferred to use low dimensional feature vector and use simple neural network structure. Among all these combinations, one good solution can be found. That is, use WPT as feature extraction algorithm with feature reduction Method II (set $H = 0.06$), get 18-point feature vectors, then use 18-18-4 MLP as the neural classifier. Thus a considerable amount of computation is avoided, while the accuracy for classification remains high.

6. Conclusion

This research has investigated the feasibility of building an automatic frog call monitoring system based on in-field acquisition of sound signals. The frog species identification has been realized in the first stage. Different algorithms including filtering and grouping are developed to identify different species. The individual frog identification of species RAUT has been performed in the second stage. Since most of the researches in the field of animal sound recognition are focused on species identification, the individual identification approach proposed herein is novel. Feature extraction algorithm using WPT with two dimensionality reduction algorithms (Methods I and II), and the neural network classifier have been synergistically integrated together to facilitate the estimation of the population within the species of interest.

References

1. W. S. Osborne, "Frogs," *Microsoft® Encarta® Online Encyclopedia 2000*, 2000, http://encarta.msn.com.
2. F. V. Brock, K. C. Crawford, R. L. Elliott, G. W. Cuperus, S. J. Stadler, H. L. Johnson and M. D. Elits, The Oklahoma Mesonet: A technical overview, *J. Atmosphere and Oceanic Tech.* **12**, 1 (1995) 5–19.
3. A. L. Mcilraith and H. C. Card, Birdsong recognition with DSP and neural networks, *Proc. IEEE Conf. Comm. Power, and Comput.*, 1995, 409–414.
4. K. Sasaki and M. Yamazaki, Vector compression of bird songs spectra in water sites by using the linear prediction method and its application to an automated Bayesian species classification, *Proc. 38th SICE Ann. Conf.*, 1999, 1083–1088.
5. A. J. Taylor, Bird flight call discrimination using machine learning, *J. Acoustical Society of America* **97**, 5 (1995) 3370–3380.
6. A. J. Taylor, G. Watson, G. C. Grigg and H. I. McCallum, Monitoring frog communities: An application of machine learning, *Proc. 8th Innovative Applications of Artif. Intell. Conf.*, 1996, 212–217.
7. Z. B. Lin, Some aspects of wavelet transform in bio sonar processing and detection, *J. Acoustical Society of America* **91**, 5 (1992) 2468–2468.
8. J. R. Deller, J. G. Proakis and J. H. Hansen, *Discrete-Time Processing of Speech Signals* (Macmillan, New York, 1993).
9. A. L. Mcilraith and H. C. Card, Birdsong recognition using backpropagation and multivariate statistics, *IEEE Trans. Sig. Processing* **45**, 11 (1997) 2740–2748.
10. J. Neter and W. Wasserman, *Applied Linear Statistical Models: Regression, Analysis of Variance and Experimental Designs*, R. D. Irwin: Homewood, Illinois, 1974.
11. H. C. Gerhardt, Acoustic properties used in call recognition by frogs, *The Evolution of the Amphibian Auditory System*, eds. Fritzsch *et al.* (John Wiley, New York, 1988) 455–483.
12. A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall, Upper Saddle River, New Jersey, 1998.
13. L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing* (Prentice Hall, Englewood Cliffs, New Jersey, 1975).
14. K. M. Fristrup and W. A. Watkins, Marine animal sound classification, *J. Acoustical Society of America* **97**, 5 (1995) 3369–3370.

15. D. K. Mellinger and C. W. Clark, Methods for automatic detection of mysticete sounds, *Marine and Freshwater Behavior and Physiology* **29**, 1 (1997) 163–181.
16. J. R. Potter and C. W. Clark, Marine mammal call discrimination using artificial neural networks, *J. Acoustical Society of America* **96**, 3 (1994) 1255–1262.
17. G. G. Yen and K. C. Lin, Wavelet packet feature extraction for vibration monitoring, *IEEE Trans. Ind. Electronics* **47**, 3 (2000) 650–667.
18. G. G. Yen and P. Meesad, Pattern classification by an incremental learning fuzzy neural network, *IEEE Trans. Syst. Man Cybernetics*, Part B: *Cybernetics*, to appear in 2001.
19. P. Meesad and G. G. Yen, Pattern classification by a neurofuzzy network: Application to vibration monitoring, *ISA Trans.* **39**, 3 (2000) 293–308.
20. M. Riedmiller and H. Braun, A direct adaptive method for faster backpropagation learning: The RPROP algorithm, *Proc. IEEE Int. Conf. Neural Networks*, 1993, 586–591.

APPENDIX K:

Acoustic Emission Data Assisted Process Monitoring

by

Gary G. Yen and Haiming Lu

ISA Transactions, to appear

ACOUSTIC EMISSION DATA ASSISTED PROCESS MONITORING

Gary G. Yen *Haiming Lu*

Intelligent Systems and Control Laboratory
School of Electrical and Computer Engineering
Oklahoma State University

Abstract

Gas-liquid two-phase flows are widely used in the chemical industry. Accurate measurements of flow parameters, such as flow regimes, are the key of operating efficiency. Due to the interface complexity of a two-phase flow, it is very difficult to monitor and distinguish flow regimes on-line and real-time. In this paper we propose a cost-effective and computation-efficient AE detection system combined with artificial neural network technology to recognize four major patterns in an air-water vertical two-phase flow column. Several crucial AE parameters are explored and validated, and we found that the density of acoustic emission events and ring-down counts are two excellent indicators for the flow pattern recognition problems. Instead of the traditional Fair map, a hit-count map is developed and a multi-layer Perceptron neural network is designed as a decision-maker to describe an approximate transmission stage of a given two-phase flow system.

Keywords: Acoustic emission, process monitoring, non-destructive testing, artificial neural network

1. INTRODUCTION

Modern engineering technology is leading to increasingly complex chemical processes with ever more demanding performance criteria. Imminent needs in optimizing the production yield and cost for global economic competition call for an even higher standard in performance reliability. A critical need with a supplementary sensory system based upon nonintrusive sensory signatures surely exists to assist the monitoring decision by operators. The research dedicated to process industry, such as the one proposed herein, will promote an ultimate enabling tool appropriate for on-line health monitoring and decision-making. To substantiate the feasibility, a generic gas-liquid two-phase vertical column is considered to validate the technology proposed.

Gas-liquid two-phase flows are defined as the flow of a mixture of two homogeneous phases, gas and liquid, through a system. Since they would aid the description of heat and mass transfer mechanisms in a system, they play a very important role and are widely used in petrochemical and chemical process industries [1]. An example of this would be the pipe flow in the gas/liquid two-phase conveying process. It has been proven that the operating efficiency of such a process is closely related to accurate measurement of flow parameters, such as flow regimes and multiple flow velocities [2]. Generally speaking, flow patterns are classified as Bubbly [3], Slug [4], Churn [5] and Annular [6]. These flow regimes typically have distinct flow characteristics and heat and mass transfer mechanisms, which are very critical for detail study in this field. Some detection techniques were applied to monitor and detect flow pattern images or measure flow velocity in previous research. Xu and Xu [7] established a mathematical model, two-value (0/1) logical back-projection filtering algorithm combined with a transmission-mode ultrasound computerized tomography system. Its purpose is to reconstruct the image of a distribution of bubbles over a 2-D cross-section of a pipe for both parallel beam scanning and fan-shape beam scanning geometry. Albusaidi and Lucas [8] proposed a technique that consists of mounting an array of 64 axially separated conductivity sensors in a vertical pipe through which an air/water mixture is flowing, to obtain the mean Cap bubble (or Taylor bubble) velocity and hence an estimation of the mean gas velocity by cross-correlation of the output signals. These detection approaches based upon ingenious sensor innovations are often local in nature, passive and labor intensive. The developed prototype instruments are heavy, expensive and fault prone. Thus it is difficult to implement these methods in a transportable, on-board, automatic, real-time and globally health assessment tool. In this paper, we propose a cost-effective, non-intrusive monitoring approach based upon acoustic emission (AE) sensors combined with an artificial neural network to study a vertical air-water, two-phase flow system and classify the four major flow patterns of this phenomenon.

Acoustic Emission is a term describing a class of phenomena, whereby transient elastic waves are generated by the rapid release of energy from localized sources within a material. AE has developed rapidly over the last two decades as a nondestructive evaluation technique and as a tool for material research. It is a highly sensitive approach for detecting active microscopic events in a material and has been successfully used in the field of monitoring the welding or crack in solid materials, such as metal, glass and ceramic under stress [9]. Some acoustic emission sensors were designed for monitoring the kinetics of chemical reactions [10]. Additionally, an acoustic emission monitoring system was built to estimate the size of suspended solids in an agitated vessel, and estimate the yield from a crystalliser [11]. In this paper, a system with AE methods was applied to detect and classify four major regimes, namely Bubbly, Slug, Churn and Annular of a vertical air-water two-phase flow.

The remainder of this paper is organized as follows. Section 2 investigates four significant flow regimes. The well-regarded Fair regime map is introduced to describe traditional regime classification technique. Section 3 discusses the principle of acoustic emission techniques and defines some important parameters for the experiment. Section 4 proposes our AE based air-water two-phase flow regime classification system. System hardware configuration is illustrated, and a multiplayer Perceptron (MLP) neural network is designed to locate the detected signal at the correct position on the AE Hit-Count map. Section 5 presents the experimental result by our designed system. Section 6 provides some concluding remarks along with pertinent observations.

2. FLOW REGIMES OF A GAS-LIQUID TWO-PHASE COLUMN

The description of a two-phase flow in pipes is highly intricate due to the various existence of the interface between the two phases. For gas-liquid two-phase flows, the variety of interface forms depends on the flow rates, phase properties of the fluid and on the inclination and the geometry of the tube. Generally, for vertical gas-liquid two-phase flows, the flow regimes are mainly determined by the phase flow rates. In this case, Bubbly, Slug, Churn and Annular are four significant regimes that can be recognized as standard patterns in the chemical industry. The characteristics of these four patterns are shown in Figure 1. Each of these four patterns has a distinguished air/water density and flow speed ratio. The calculation of the flow rates is required to ensure that all the flow patterns could be observed. In order to obtain all the required flow rates with the equipment, the flow regime map developed by Fair [12] was used. The map, shown in Figure 2, is a plot of a Martinelli parameter [13] X_{tt}, given by

$$X_{tt} = \left(\frac{1-x}{x}\right)^{0.9} \times \left(\frac{\rho_g}{\rho_l}\right)^{0.5} \times \left(\frac{\mu_l}{\mu_g}\right)^{0.1} \tag{1}$$

versus the total mass velocity, G_T, defined as

$$G_T = \frac{\dot{m}_g + \dot{m}_l}{S}, \tag{2}$$

where quantity x is defined as the mass fraction of the gas phase in the two phase mixture and is given by

$$x = \frac{\dot{m}_g}{\dot{m}_g + \dot{m}_l}. \tag{3}$$

Other parameters involved are defined as

$\dot{m}_g$ - gas flow rate, in lb/s;

$\dot{m}_l$ - liquid flow rate, in lb/s;

ρ_g - gas density, in lb/ft^3;

ρ_l - liquid density, in lb/ft^3;

μ_g - gas viscosity, in lb/ft-s;

μ_l - liquid viscosity, in lb/ft-s; and

S - area of cross section of the pipe, in ft^2.

In our experiment, the phase flow rates can be measured by given meters. The pipe radius is known, and the density and viscosity values of air and water are given in standard look up tables. Therefore, we can map the obtained data to the exact position on the fair map, which provides a reference to the flow regimes of a AE classification system.

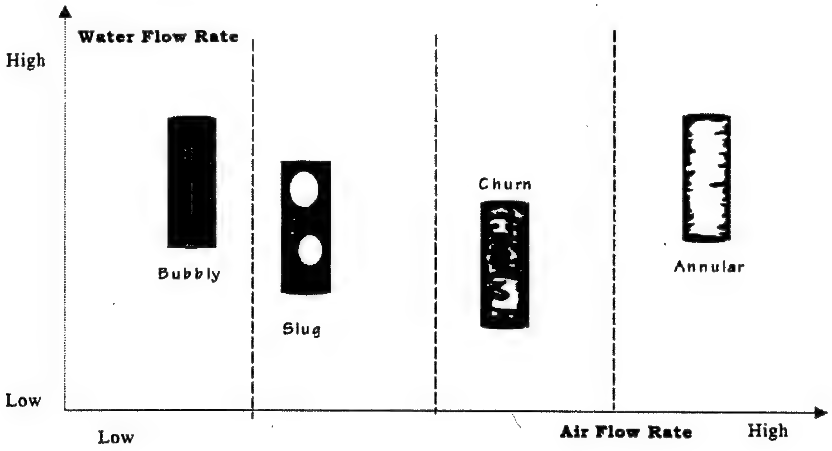


Figure 1 Water/air flow ratio of four major two-phase vertical flow patterns

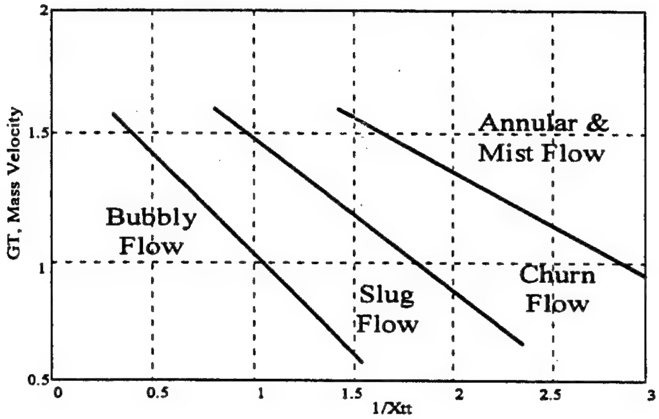


Figure 2 Fair map—four major regimes

3. ACOUSTIC EMISSION TECHNIQUE

Acoustic emission testing is a powerful method for examining the behavior of materials in which a transient elastic wave is generated by a rapid release of energy. During the AE test, the sensors on the test piece produce any number of transient signals. A signal from a single, discrete event is known as a burst-type signal. This type of signal has a fast rise time and a slower decay, as illustrated in Figure 3. Burst-type signals vary widely in shape, size and rate of occurrence, depending on the structure and test conditions. Several parameters of an AE signal need to be defined as following:

a) AE threshold – A predefined value to indicate the occurrence of an AE hit and a number of AE counts.
b) AE hit (event) – Occurs when the amplitude value of the sensor output signal is higher than the predefined threshold.
c) AE signal duration – The period between hit starting and ending points.
d) AE ring-down count – The number of the threshold-crossing pulses.

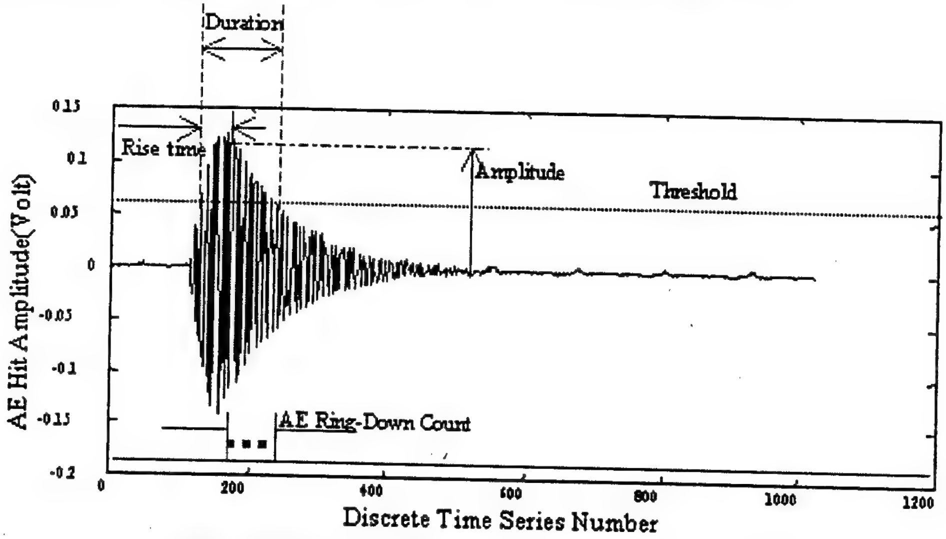


Figure 3 Standard AE event (hit) signal in time domain

Figure 4 shows the frequency spectrum of an AE signal.

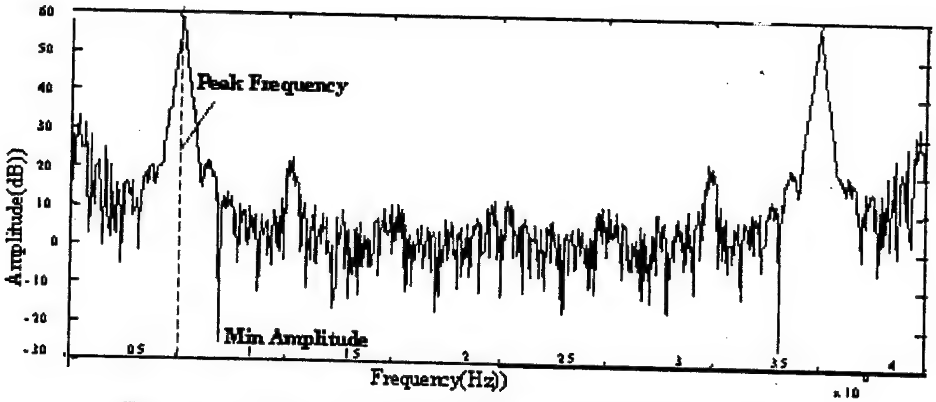


Figure 4 Standard AE event (hit) signal in frequency domain

Since AE signals in our experiment are of relatively short durations (less than 1 msec), reach maximum amplitude early in the signal (always assume 0) and decay exponentially, as shown in Figure 3, we can calculate the sensor output as:

$$V(t) = V_0 \, e^{-rt} \sin wt \qquad (4)$$

where:

$V(t)$ - output voltage of sensor;
V_0 - initial signal amplitude;
r - decay constant (>0);
t - time; and
w - signal frequency.

Since Threshold voltage V^* has been set up, we can count the number of times the sensor voltage exceeds it. This technique is known as ring down counting. For the signal represented by Equation (4), the number of counts (N) to the nearest integer is given by

$$N = \frac{t^*}{2\pi / w} = \frac{w}{2\pi r} \ln \frac{V_0}{V^*} \tag{5}$$

where

$$V^* = V_0 e^{-rt^*}; \text{ and } t^* = \frac{1}{r} \ln \frac{V_0}{V^*}. \tag{6}$$

For the given air-water two-phase flow classification problem, we obtain the average values of each second for those related parameters introduced above. Table 1 shows the comparison results for the four major patterns.

Table 1 AE parameter values of four major flow regimes

	Bubbly	**Slug**	**Churn**	**Annular**
Average number of AE hits occurs in one second	5 (0~58)	83 (25~150)	185 (134~243)	39 (4~92)
Average number of AE counts occurs in one second	87 (0~32)	749 (17~4923)	8192 (2487~14236)	30521 (13510~44673)
Average value of amplitude for the AE hits occurs in one second (dB)	61.56 (60.17~61.92)	60.59 (59.81~60.7)	61.43 (59.93~61.65)	61.13 (60.11~61.96)
Average Rise time for the AE hits occurs in one second (us)	17.7 (1~59)	20.49 (1~429)	26.47 (1~711)	35.37 (1~717)
Average Duration time for the AE hits occurs in one second (us)	75 (1~264)	127.9 (1~8468)	314.6 (1~43012)	430.6 (1~132259)

*** The values in () are the value range of the given AE parameters in one second**

From data analysis summarized in Table 1, we can see that the number of AE ring-down counts, occurring in one second, is the most reliable indicator for the given pattern classification problem. To ensure the reliability of the final classification result, we also combine the AE hit (event) number, occurring in one second, to be another indicator. By mapping one-second data to a point on the Hit-Count map, we can classify the four major flow regimes, which will be shown in Section 5. For a more complicated gas-liquid vertical column to be monitored, all features discussed above can be integrated into the decision-making process.

4. AE CLASSIFICATION SYSTEM

The hardware configuration of the proposed real-time AE air-water two-phase flow classification system is shown in Figure 5. The system includes data acquisition, signal processing, data analysis and decision-making. In Figure 5, sensor A is an AE sensor which is attached on the pipe for detecting AE signals that occurs in flows, and sensor B is same type of piezoelectric AE sensor, which is located near the sensor A and designated for detecting background noise. Related AE hardware parameters are listed in Table 2. The output data from AE sensors are amplified and filtered before the A/D conversion. In data analysis and decision making parts, the input of the MLP neural network is the discrete data from the AEDSP card manufactured by Physical Acoustics Corporation. The network output determines the current position on the Hit-Count map, which can be the indicator of the decision-making Graphic User Interface (GUI) software.

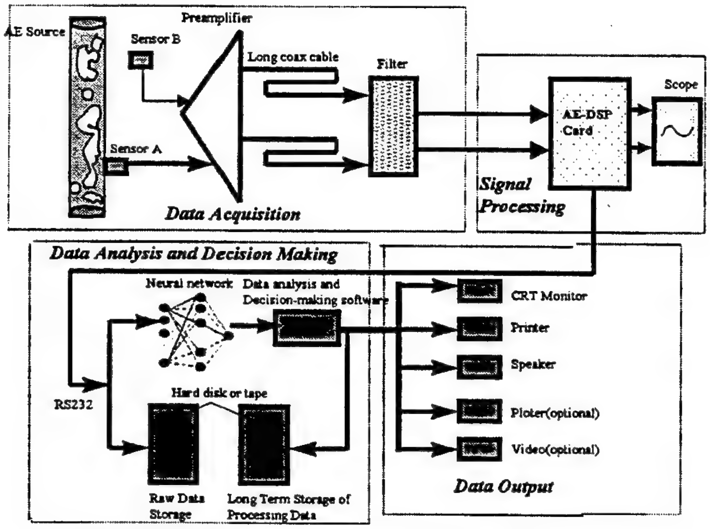


Figure 5 Configuration of experimental AE detection system

Table 2 AE hardware parameter values

AE Sensor Resonant Frequency	*150KHz*
Sampling Rate of DSP Board	*1MHz*
Gain of Amplifier	*40dB*
Time Window of Each AE Hit	***1024 points*** ($256\mu s$)
Threshold Voltage	***0.0586V*** ($35dB$)

5. EXPERIMENTAL RESULTS

In the experiment, we designed a continuous regime-changing process which includes 12 steady states and 11 transient states during 20 minutes data acquisition time. Each steady state keeps a pair of typical air and water flow rates (Figure 6) for about 30 seconds and then transfers to another steady state. The state between two steady states is the transient state, which has unstable phase flow rates. The phase flow rates of all the steady and transient states were recorded to generate our reference Fair map (Figure 7). Meanwhile, the AE hit and count number for each second-period are also measured and stored during the entire process.

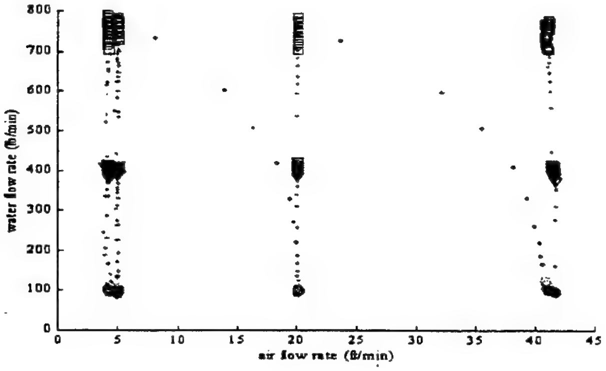


Figure 6 Air and water flow rates for steady states and transient states

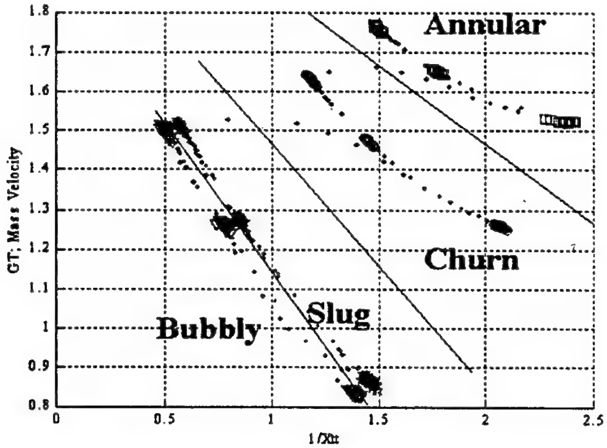


Figure 7 Reference Fair map for the four major Regimes

Figure 8 shows the corresponding AE Hit-Count map, and each marked point represents the summation AE hit number and count number for one second. From this Hit-Count map, we can see that regime "Annular" is clearly separated from the other three regimes. Regimes "Churn" and "Slug" are also well separated, although there exists a small overlapping between these two regimes. The regime "Bubbly" and "Slug" can not be linearly separated since the overlapping area between them is not trivial. However, we can train a nonlinear classifier, such as MLP neural network to classify them, because the majority of these two regimes are separable.

In our experiment, we use a two-hidden-layer neural network with 10 neurons in each hidden layer. We randomly select 600 data points for training and the other 600 data points for testing. Levenberg-Marquardt with Bayesian regularization algorithm is applied, and the learning rate is set to be 0.1. The training epoch is 1,000, and the stopping mean square error is 1e-5. Figure 9 (a) and (b) show the training result Hit-Count map and the training target Hit-Count map. Figure 10 (a) and (b) show the testing result Hit-Count map and testing target Hit-Count map.

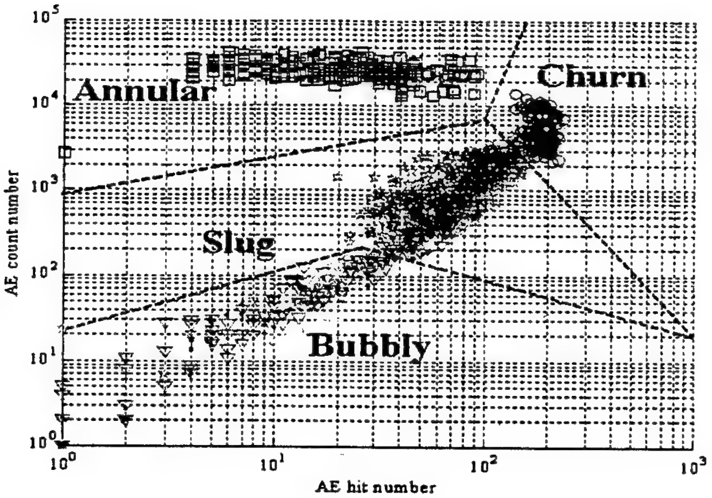
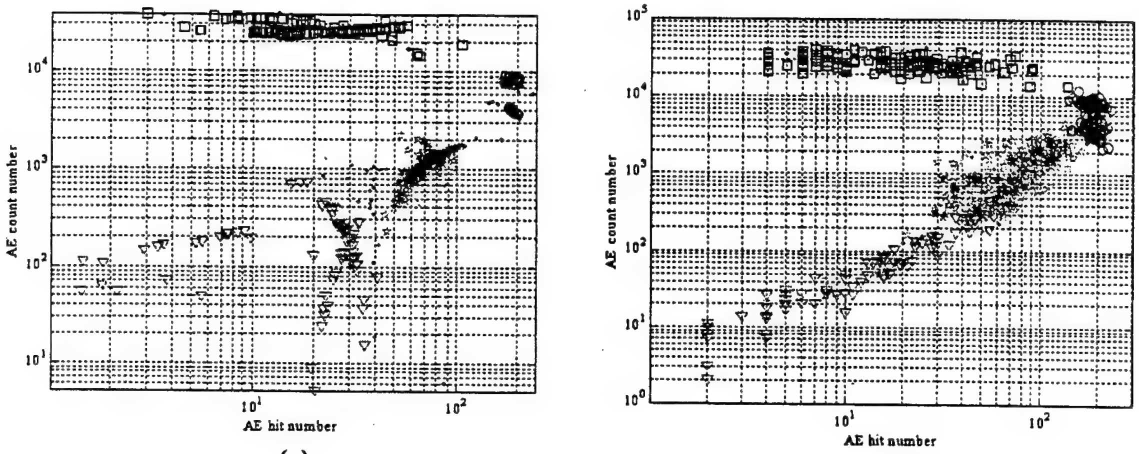


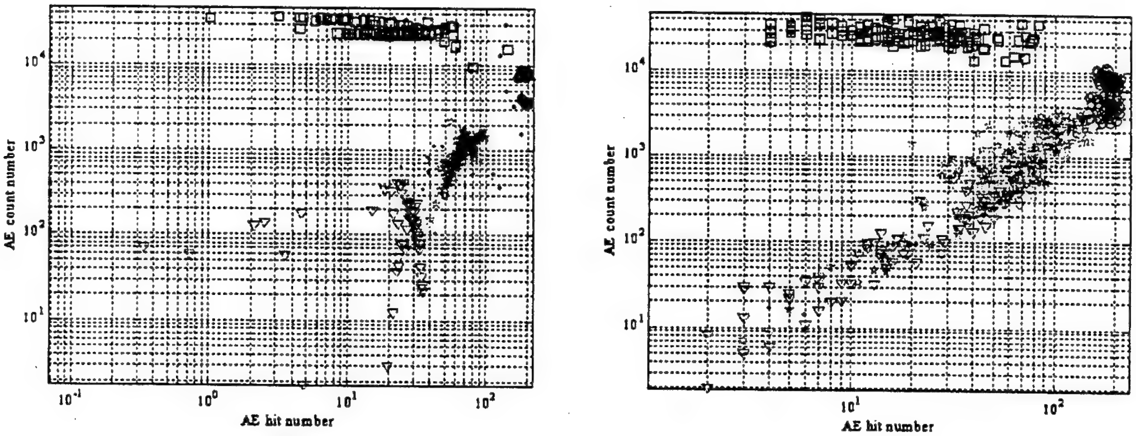
Figure 8 AE Hit-Count map corresponding to the Fair map in Figure 7



(a)

(b)

Figure 9 (a) Neural network training result Hit-Count map and (b) Training target Hit-Count map



(a)

(b)

Figure 10 (a) Neural network testing result Hit-Count map and (b) Testing target Hit-Count map

Comparing the training and testing result Hit-Count maps with the target maps in Figures 9 and 10, we can see that the overlapping areas between adjacent regimes have disappeared due to the clustering character by the neural network. In this case, the designed neural network can improve the classification performance by reducing the misclassification rate for the given air-water two-phase flow regime classification problem. The output of the neural network is transferred to the decision-making and Graphic User Interface software. This software is designed to have the following functions: 1) indicate the current flow regime; 2) control the starting and stopping of the data acquisition process by user; and 3) perform database management for history record inquiry. Figure 11 shows the GUI software. The developed interface allows the operator to visually monitor the process to facilitate the expert decision-making. By integrating with concepts, such as distributed virtual instrumentation, the operator not only can remotely monitor the process, but activate the control law for reconfiguration via Ethernet [14].

6. CONCLUSION

The application of one of the NDT techniques (i.e., AE) and Neural Networks on vertical air-water two-phase flow pattern recognition problems was proposed and discussed. In this study, several AE parameters were extracted from four major two-phase flow pattern signals, and the results were discussed. AE hits (events) and Ring-Down Counts density can be combined as a stable and excellent indicator to describe flow patterns accurately. They form the input stream of multi-layer perceptron neural network. After training the network, the system output can tell the continuous flow stage (including four major patterns and transient states) on-line and real time. This AE combined NN detection system may be easily transferred to other gas/liquid two-phase flow regime classification problems, such as saturated steam flow, which is widely related to different industrial processes for heat energy transfer, power source, sanitary flushing, and etc. Some common flow regimes for saturated steam flows are uniform density regime, annular regime, slug regime, and asymmetric density regime. While not demonstrated, we fully believe that the proposed acoustic emission monitoring system will work equally well for saturated steam and other similar two-phase flow systems.

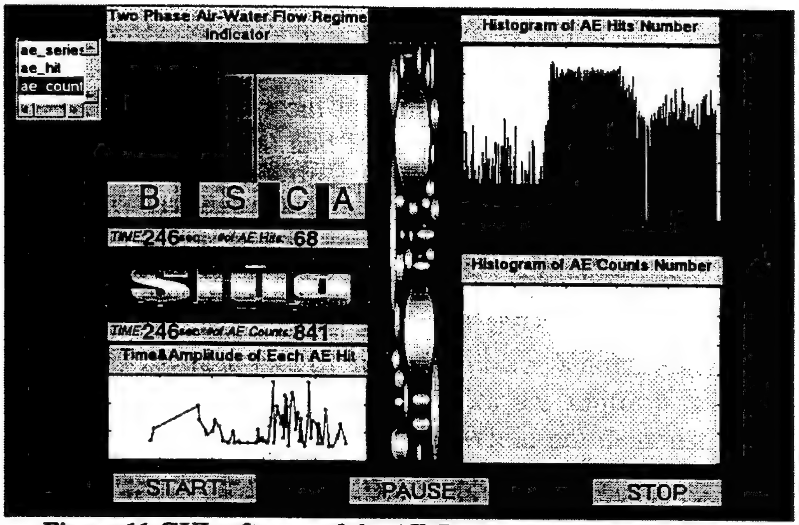

Two Phase Air-Water Flow Regime Indicator
ae_series
ae_hit
ae count
B
S
C
A
TIME 246 sec #of AE Hits: 68
TIME 246 sec #of AE Counts: 841
Time&Amplitude of Each AE Hit
Histogram of AE Hits Number
Histogram of AE Counts Number
START
PAUSE
STOP

Figure 11 GUI software of the AE flow regime detection system

REFERENCES

[1] Hewitt, G. F., *Measurement of Two Phase Flow Parameters,* London: Academic, 1978.

[2] Galegar, W. C., Stovall, W. B. and Huntington, R. L., "More data on two-phase vertical flow," *Petroleum Refineries*, **33**, pp. 208-219, 1954.

[3] Rhodes, E., "Gas-liquid flow," (on video tape), *Heat Transfer & Fluid Flow Services, Atomic Energy Research Establishment*, London: Harwell, 1982.

[4] Fernandes, R. C., Semiat, R. and Dukler, A. E., "Hydrodynamic model for gas-liquid slug flow in vertical tubes," *AIChE Journal,* **29**, pp. 981-995, 1983.

[5] Taitel, Y., Bornea, D. and Dukler, A. E., "Modelling flow pattern transmissions for steady upward gas-liquid flow in vertical tubes," *AIChE Journal,* **26**, pp. 345-359, 1980.

[6] Hewitt, G. F., "Disturbance waves in annular two-phase flow", *Proceedings of Industrial Mechanical Engineering*, **18**, pp. 142-149, 1969.

[7] Xu, L. J. and Xu, L. A., "Ultrasound tomography system used for monitoring bubbly gas/liquid two-phase flow", *IEEE Transactions on Ultrosonics, Ferroelectrics, and Frequency Control*, **44**, pp. 67-74, 1997.

[8] Albusaidi, K. H. and Lucas, G., " Measurement of multiple velocities in multiphase flow", *IEEE Colloquium on Advances in Sensors for Fluid Flow Measurement,* **12**, pp 1-4, 1995.

[9] Bassim, M. N., Dudar, M.P., Rifat, R. and Roller, R., "Application of acoustic emission for nondestructive evaluation of utility inductive reactors", *IEEE Transactions on Power Delivery*, **8**, pp. 281-284, 1993.

[10] Bang, S. W., Lec, R. M., Genco, J. M. and Ransdell, J. C., "Acoustic emission chemical sensor" *IEEE Proceedings of Ultrasonic Symposium, 1993*, **1**, pp. 439-443, 1993.

[11] Bouchard, J. G., Payne, P. A. and Szyszko, S., "Non-invasive measurement of process states using acoustic emission techniques coupled with advanced signal processing," *IChemE Transactions*, **72**, pp. 20-25, 1994.

[12] Fair, J.R., "What you need to design thermosiphon reboilers," *Petroleum Refineries*, **39**, pp.105-116, 1960

[13] Martinelli, R. C., Nelson, D. B., Schenectady, N. Y., "Prediction of pressure drop during forced-circulation boiling of water," *ASME Transaction*, **70**, pp. 695-704, 1948

[14] Yen, G. G., "Sensory-based expert monitoring and control," *Proceedings of 2nd International Conference on Information Fusion*, pp. 953-959, 1999.

APPENDIX L:

Overview Charts

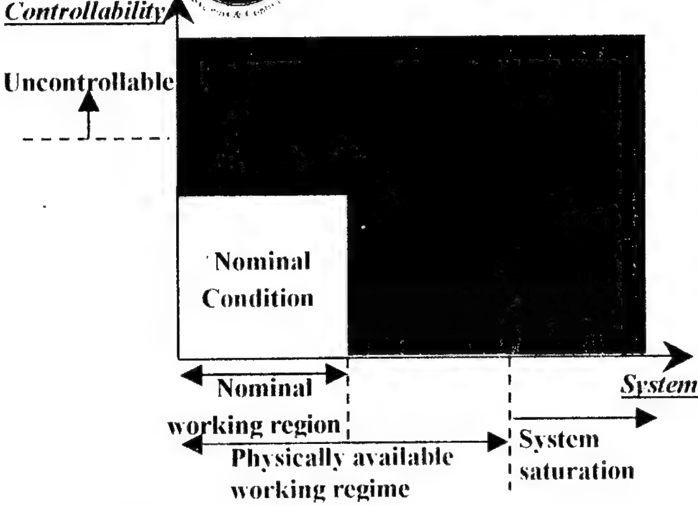
by

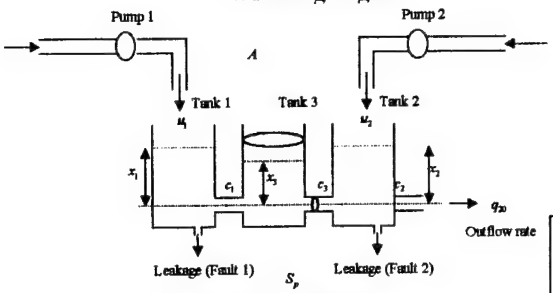
Gary G. Yen and Graduate Students

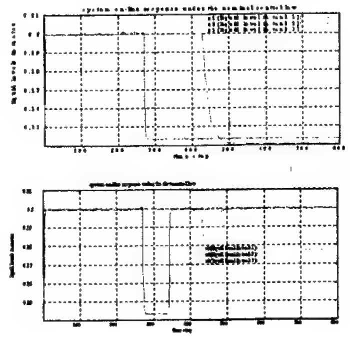
ON-LINE FAULT DIAGNOSIS AND ACCOMMODATION FOR SYSTEM FAILURES

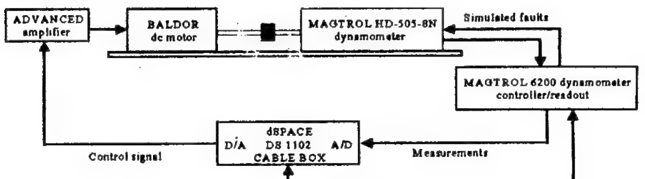
Liangwei Ho, Ph.D., Fall 2000

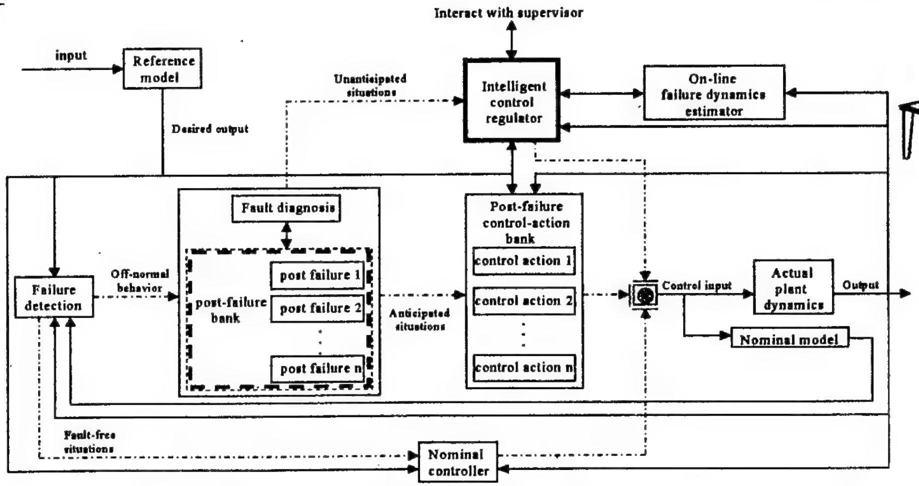
ABSTRACT : *The objective of this study is to to investigate on-line fault tolerant control problems for unanticipated catastrophic system failures in the most general formulation. The necessary and sufficient conditions for system on-line stability under catastrophic failures have been derived based upon discrete-time Lyapunov stability theory. An on-line fault accommodation control framework is proposed to deal with the desired trajectory-tracking problem for systems suffering from various unknown and unanticipated catastrophic failures. Artificial Neural Network is suggested as the on-line estimator to learn and approximate the unknown system failure dynamics. Effective control signals to accommodate the dynamic failures are then computed through the realization of the neural network estimator for the unknown failures based only upon the partially available information of the faults. A more complete architecture for intelligent fault diagnosis and accommodation has also been presented by incorporating the developed intelligent fault tolerant control technique with a cost-effective fault detection scheme and a multiple-model based failure diagnosis process to efficiently handle the false alarms, the accommodation of the anticipated failures, and to reduce the unnecessary control effort and computational complexity in on-line situations.*



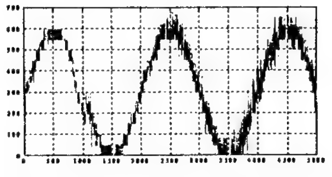




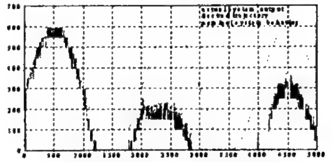




An architecture of multiple-model based fault diagnosis and accommodation



Real-time system response under the alternative corrective control law



Real-time system response under nominal controller

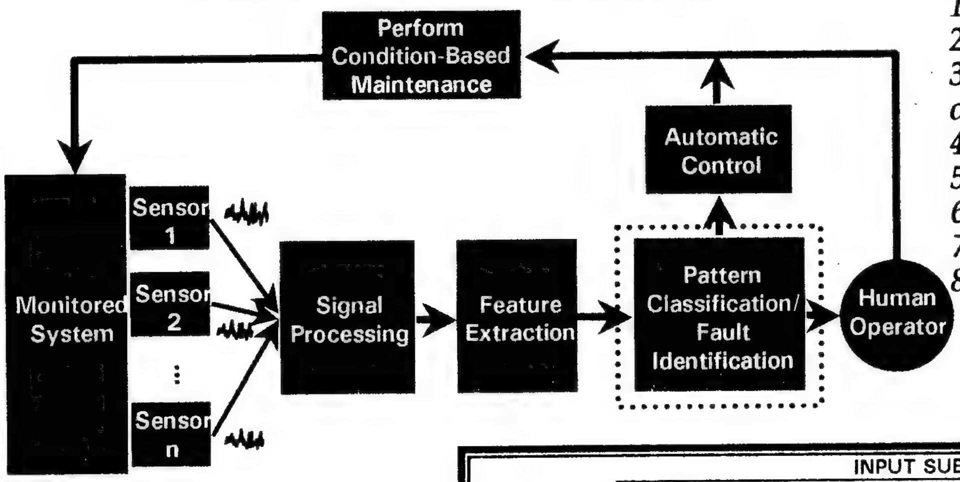
PATTERN CLASSIFICATION BY AN INCREMENTAL LEARNING FUZZY NEURAL NETWORK

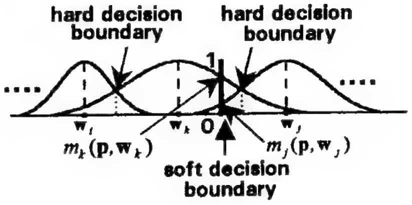
Phayung Meesad, M.S., Fall 1998

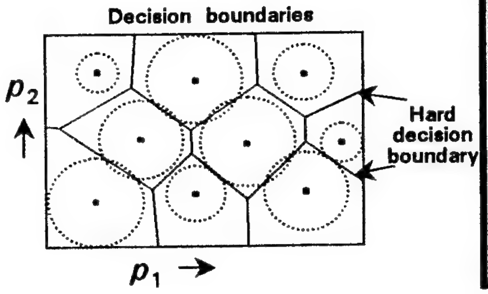
ABSTRACT: This study is to develop a new methodology of pattern classification suitable for machine health monitoring systems. Features include:

1) using fewer tuning parameters,
2) being a nonparametric classifier,
3) being a fast, on-line, real-time, one-pass, and incremental learning classifier,
4) ability to build nonlinear decision boundaries,
5) ability to classify overlapping classes,
6) a hybrid supervised and unsupervised learning,
7) ability to provide both soft and hard classification decisions, and
8) ability to detect and learn new classes in an operating mode.

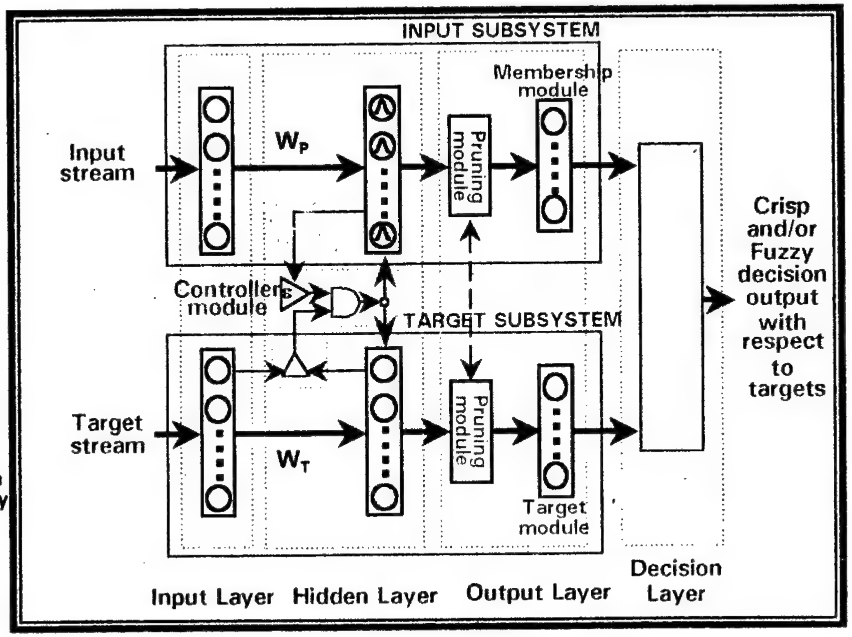
machine health monitoring system

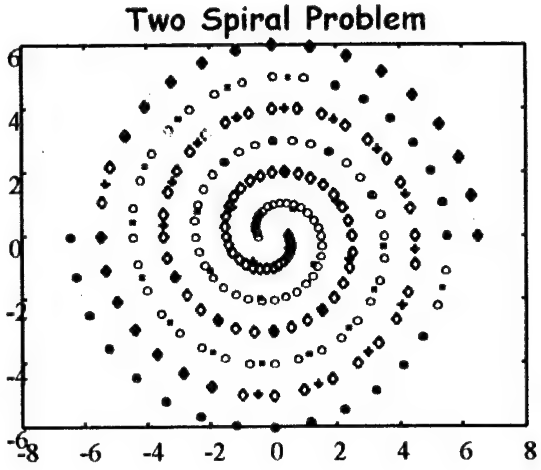






ILFN classifier





GENETIC ALGORITHM FOR MULTIOBJECTIVE OPTIMIZATION

Haiming, Lu Ph.D. Candidate

ABSTRACT : Genetic Algorithms (GA's), with the capability of parallel searching, has been well adopted to solve Multiobjective Optimization problems since a family of "acceptable" solutions, so called Pareto set, can be identified by different individuals through the evolution process. However, most of existing Multiobjective Genetic Algorithms (MOGAs) have the difficulty in dealing with the trade-off characters between uniformly distributing computation resource to high dimensional searching space and genetic drifting phenomenon. In this study, a new Rank-density based Genetic Algorithm (RDGA) is proposed to synergistically integrate several features of previous MOGAs in a unique way. Automatic ranking, adaptive density calculating, diffusion and elitism scheme, and forbidden region technique are applied in RDGA. RDGA clearly outperforms the selected three MOGAs on a Hierarchical Genetic Algorithm based Radial Basis Function Neural Network (HGA-RBFNN) design problem in terms of keeping the diversity of the individuals along trade-off surface, tending to extend Pareto front to new area, and finding well-approximated Pareto optimal set.

MULTIOBJECTIVE OPTIMIZATION

A general multiobjective function optimization problem can be described as a vector function **f** that maps a set of *m* parameters (*decision variables*) to a set of *n* objectives

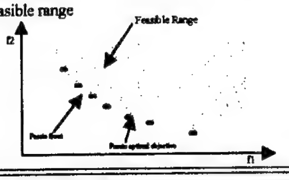
$$\text{min/max} \quad \mathbf{y} = \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots\ldots, f_n(\mathbf{x}))$$

$$\text{subject to} \quad \mathbf{x} = (x_1, x_2, \ldots\ldots x_m) \in X$$

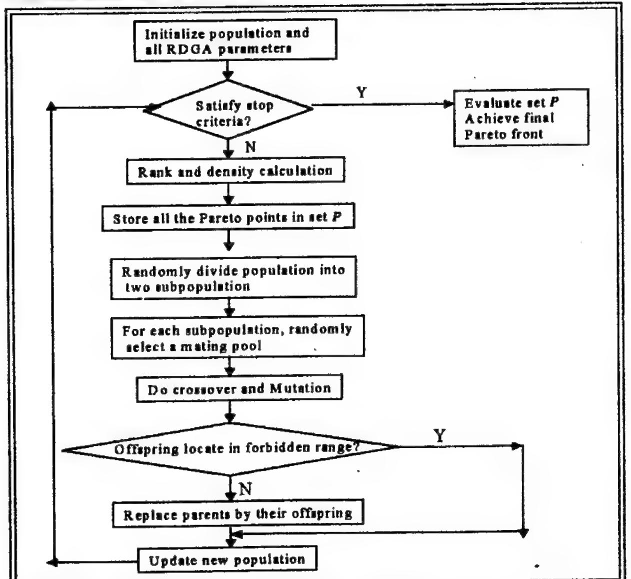
$$\mathbf{y} = (y_1, y_2, \ldots\ldots y_m) \in Y$$

where **x** is called *decision vector* which includes *m decision variables*, X is the *parameter space*, **y** is the *objective vector* which includes *n objectives*, and Y is the *objective space*.

Pareto set is a set of acceptable solutions which cannot be dominated by other solutions in the feasible range

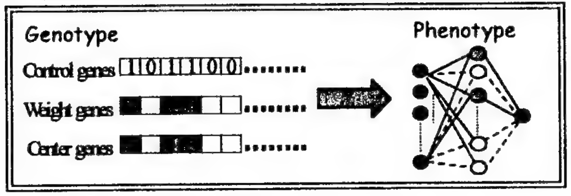


RANK-DENSITY GENETIC ALGORITHM

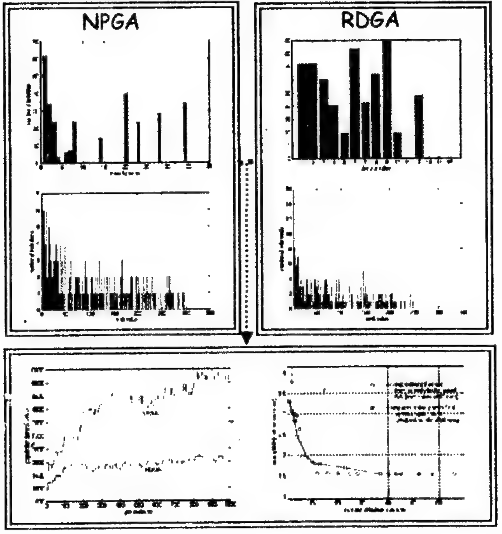

Initialize population and all RDGA parameters
Satisfy stop criteria?
Y
N
Evaluate set *P*
Achieve final Pareto front
Rank and density calculation
Store all the Pareto points in set *P*
Randomly divide population into two subpopulation
For each subpopulation, randomly select a mating pool
Do crossover and Mutation
Offspring locate in forbidden range?
Y
N
Replace parents by their offspring
Update new population

APPLICATION – NEURAL NETWORK DESIGN

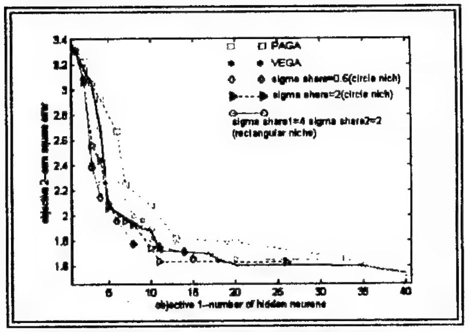
Hierarchical Genetic Algorithm based Radial Basis Function Neural Network (HGA-RBFNN)



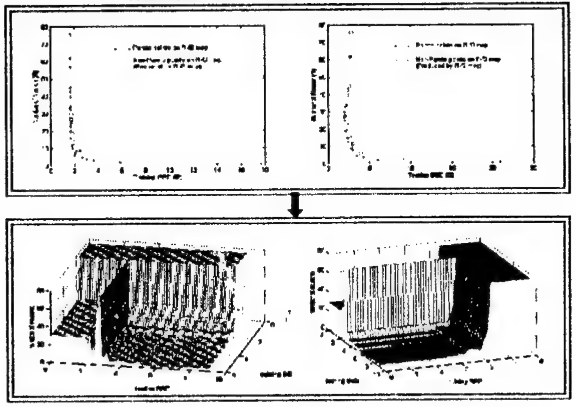
Comparison result of HGA-RBFNN design by NPGA and RDGA



Comparison result of HGA-RBFNN design by selected MOGAs (PAGA, VEGA and NPGA)



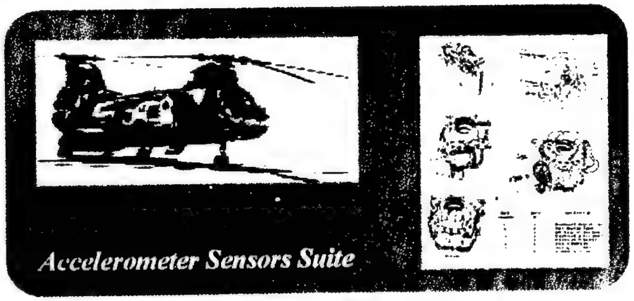
HGA-RBFNN design by RDGA – Three objective optimization



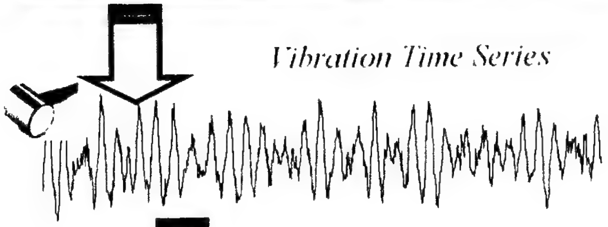
WAVELET ?ACKET FEATURE EXTRACTION FOR VIBRATION MONITORING

Kuo-Chung Lin, M.S., Fall 1998

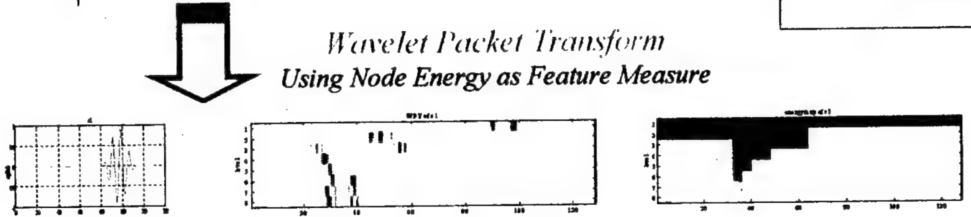
***ABSTRACT** : Wavelet Packet Transform is used to extract arbitrary time-frequency information in vibration time series data. The extracted coefficients are post-processed via Fisher's criterion to obtain a reduced dimensional feature vector for classification of vibration signatures. Under noisy environments, the wavelet packet based approach shows promising results than Fourier based approaches.*



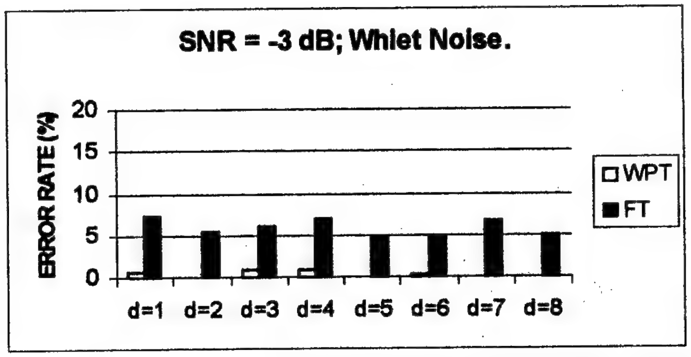
Accelerometer Sensors Suite



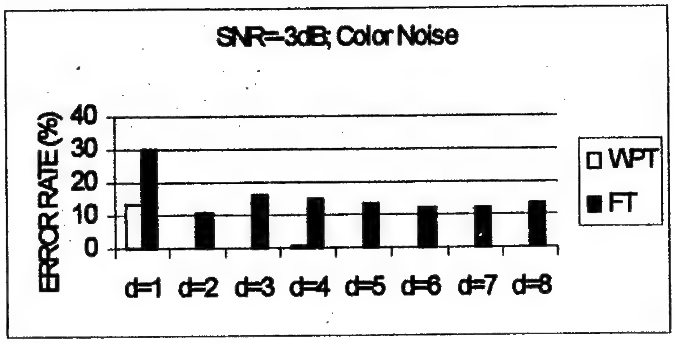
Vibration Time Series



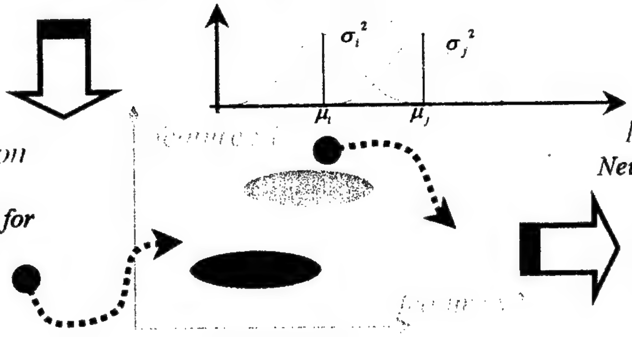
Wavelet Packet Transform
Using Node Energy as Feature Measure



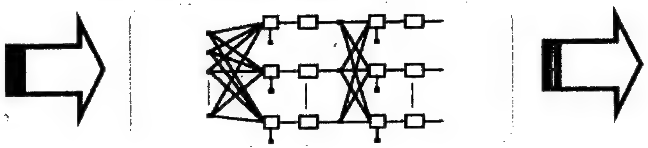
SNR = -3 dB; Whlet Noise.



SNR=-3dB; Color Noise



Feature Selection
Using
Fisher's Criterion for
Classification



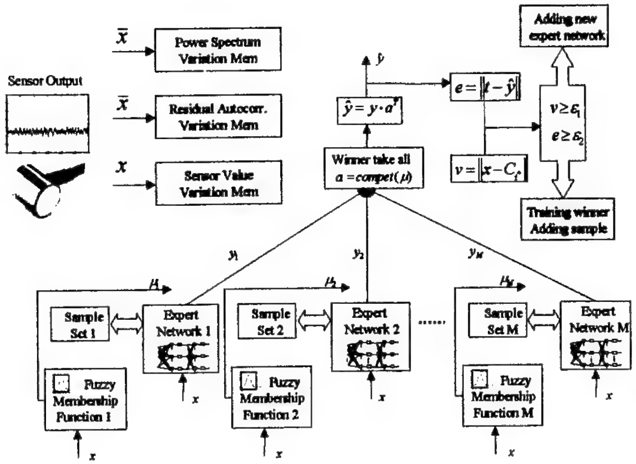
Pattern Classifier
Neural Network Classifier

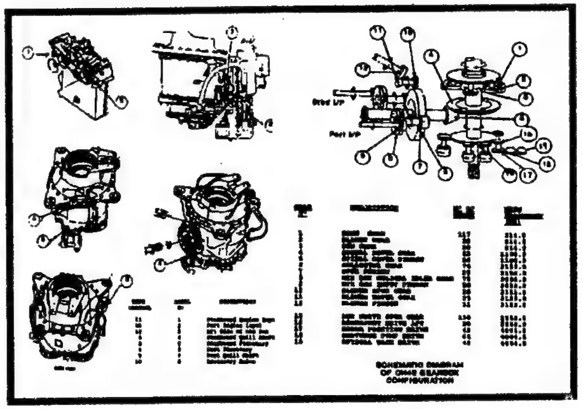
Class Label Report
Machine Current Health State

WINNER TAKE ALL EXPERT NETWORK FOR SENSOR VALIDATION

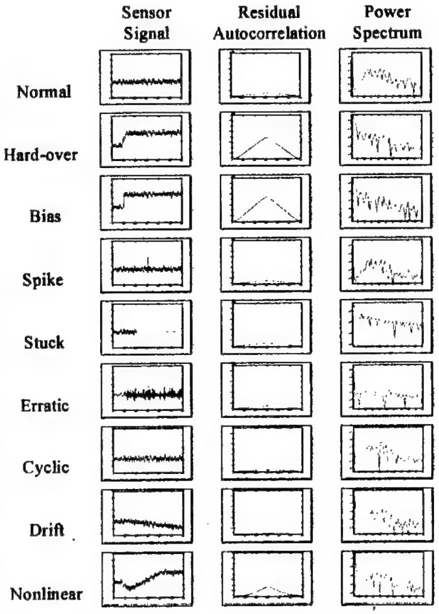
Wei Feng, M.S., Spring 2000

ABSTRACT : *The proposed Winner Take All Experts (WTAE) network is based on a 'divide and conquer' strategy. It employs a growing fuzzy clustering method to divide a complicated problem into a series of simpler sub-problems and assign an expert to each of them. After the sensor tracking, the outputs from the estimator and the real sensor value are compared both in the time domain and the frequency domain. Three fault validation gates are used to detect the sensor failure. In the decision stage, the intersection of three fuzzy sets accomplishes a decision level fusion, which indicates the confidence level of the sensor health status.*

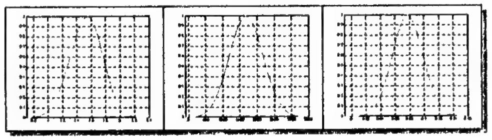




tracking



validation



Sensor States	Sensor Value Validation Gate	Residual Autocorrelation Validation Gate		Power Spectrum Validation Gate		Fuzzy Intersection
		Manhattan Distance	Confidence Level	Manhattan Distance	Confidence Level	
Normal	0.9046	1.6723e+003	0.9832	0.0771	0.9447	0.9046
Hard-over	2.8921e-014	6.1907e+005	1.0613e-008	2.7423	1.0854e-024	1.0854e-024
Bias	7.1865e-016	7.2108e+005	1.0623e-008	3.1193	1.0854e-024	1.0854e-024
Spike	7.1858e-016	1.6460e+003	0.9989	0.0822	0.9721	7.1858e-016
Stuck	0.8460	5.6381e+003	1.0591e-008	1.5572	1.0854e-024	1.0854e-024
Erratic	0.9268	1.5550e+003	1.0613e-008	0.4496	1.0854e-024	1.0854e-024
Cyclic	0.4501	4.4328e+004	1.0613e-008	0.4698	1.0854e-024	1.0854e-024
Drift	1.2242e-006	6.1678e+004	1.0613e-008	0.8432	1.0854e-024	1.0854e-024
Nonlinear	5.2260e-014	3.1871e+004	1.0613e-008	1.1034	1.0854e-024	1.0854e-024
Correction Rate	66.67%	88.89%		88.89%		100%

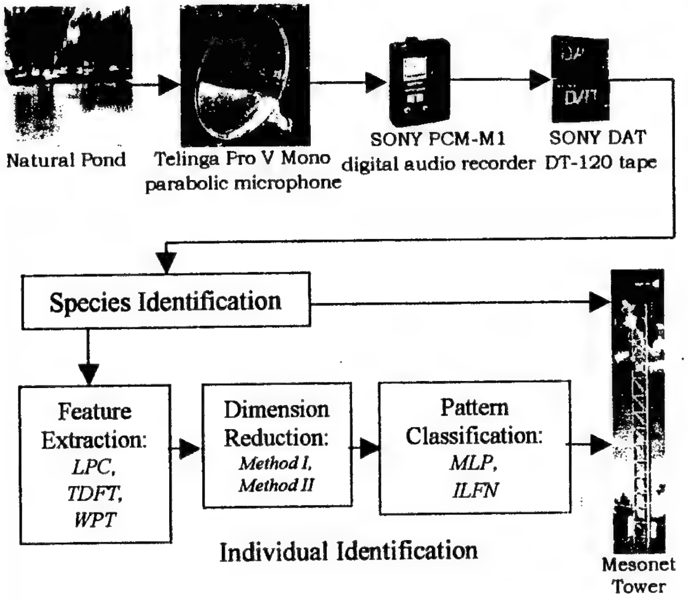
performance

AUTOMATED FROG CALLS MONITORING SYSTEM: A MACHINE LEARNING APPROACH

Qiang Fu, M.S., Fall 2000

***ABSTRACT** : The proposed monitoring system deployed one microphone to record frog calls in the field continuously. Sound signals were stored in digital audiotape first with digital audio recorder and then transmitted into a PC with WAVE file format. Species identification was performed first with the proposed filtering and grouping algorithm. Individual identification was performed in the second stage. Digital signal pre-processing, feature extraction, feature vector dimension reduction and pattern recognition were performed step by step in this stage. Simulation results shows the promising future of this monitoring system.*

Monitoring System Architecture



Natural Pond

Telinga Pro V Mono parabolic microphone

SONY PCM-M1 digital audio recorder

SONY DAT DT-120 tape

Species Identification

Feature Extraction: *LPC, TDFT, WPT*

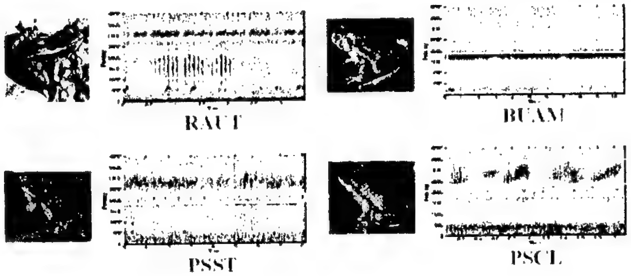
Dimension Reduction: *Method I, Method II*

Pattern Classification: *MLP, ILFN*

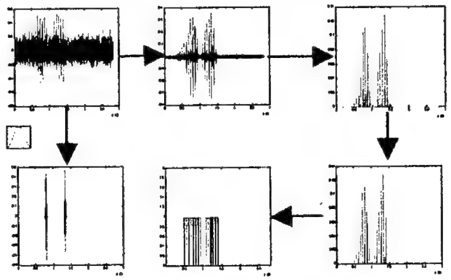
Individual Identification

Mesonet Tower

Spectrogram of Four Frog Species



Filtering & Grouping Algorithm



Simulation Results for Species Identification

RAUT and PSCL: nearly 100% accuracy;

PSST and BUAM: 100% accuracy.

Simulation Results for Individual Identification

LPC N=9		N-N-4
	Mean	0.5068
	Var.	0.0104

WPT	N		N-N-4	N-10-10-4
Method I d=6	19	Mean	0.6827	0.6809
		Var.	0.0097	0.0105
Method II H=0.06	18	Mean	0.7118	0.7164
		Var.	0.0068	0.0100
Method I d=8	31	Mean	0.6609	0.6891
		Var.	0.0095	0.0102
Method II H=0.1	33	Mean	0.7218	0.6955
		Var.	0.0076	0.0150

APPENDIX E:

Combined Numerical and Linguistic Knowledge Representation and Its Application to Medical Diagnosis

by

Phayung Meesad and Gary G. Yen

Submitted to
IEEE Transactions on Systems, Man and Cybernetics

Combined Numerical and Linguistic Knowledge Representation and Its Application to Medical Diagnosis*

Phayung Meesad and Gary G. Yen

Intelligent Systems and Control Laboratory
School of Electrical and Computer Engineering
Oklahoma State University
Stillwater, OK 74078

Abstract—In this study, we propose a novel hybrid intelligent system (HIS) which provides a unified integration of numerical and linguistic knowledge representations. The proposed HIS is a hierarchical integration of an incremental learning fuzzy neural network (ILFN) and a linguistic model, i.e., fuzzy expert system (FES), optimized via the genetic algorithm (GA). The ILFN is a self-organizing network with the capability of fast, one-pass, online, and incremental learning. The linguistic model is constructed based on knowledge embedded in the trained ILFN or provided by the domain expert. The knowledge captured from the low-level ILFN can be mapped to the higher-level linguistic model and vice versa. The GA is applied to optimize the linguistic model to maintain high accuracy, comprehensibility, completeness, compactness, and consistency. The resulted HIS is capable of dealing with low-level numerical computation and higher-level linguistic computation. After the system being completely constructed, it can incrementally learn new information in both numerical and linguistic forms. To evaluate the system's performance, the well-known benchmark Wisconsin breast cancer data set was studied for an application to medical diagnosis. The simulation results have shown that the proposed HIS perform better than the individual standalone systems. The comparison results show that the linguistic rules extracted are competitive with or even superior to some well-known methods.

Index terms—ILFN, fuzzy expert system, GA, hybrid intelligent system, pattern classification, decision support system, medical diagnosis, Wisconsin breast cancer database

I. INTRODUCTION

Conventional medical diagnosis in clinical examinations highly relies upon physicians' experience. Physicians intuitively exercise knowledge obtained from previous patients' symptoms. In everyday practice, the amount of medical knowledge grows steadily such that it may become difficult for physicians to keep up with all the essential information gained. For physicians to quickly and accurately diagnose a patient, there is a critical need in employing computerized technologies to assist in medical diagnosis and accessing to the information related. Computer-assisted technology is certainly helpful for inexperienced physicians in making medical decisions as well as for experienced physicians in supporting complex decisions. Computer-assisted technology has become an attractive tool to help physicians in retrieving the medical information as well as in making decisions in medical diagnosis [1]-[7].

A number of medical diagnostic decision support systems (MDSS) based on computational intelligence methods have been developed to assist physicians and medical professionals. Some medical diagnosis systems based on computational intelligence methods use

* This work was supported in part by the Royal Thai Government and the U.S. Air Force Office of Scientific Research under grant F49620-98-1-0049.

expert systems (ESs) [8]-[11], fuzzy expert systems (FESs) [12]-[15], artificial neural networks (ANNs) [16]-[19], and genetic algorithms (GAs) [20]-[22]. ESs and FESs, used symbolic and linguistic knowledge, respectively, are well recognized as applications in medical diagnosis because their decisions are easy to understand by physicians and medical professionals. However, the development of an ES or a FES for medical diagnosis is not a trivial task. It demands an intensive and iterative process from medical experts who may not be readily available. ANNs have been employed to learn numerical data recorded from sensory measurements or images. After being trained, ANNs keep knowledge in numerical weights and biases that are often regarded as a black box scheme. This knowledge is difficult for physicians or medical professionals to understand the underlying rationale. Recently, the numerical weights of ANNs can be translated to symbolic/linguistic rules by using rule extraction algorithms [23]-[29]. Symbolic/linguistic rules extracted are then used as a knowledge base for an ES or a FES to support the physicians in making decisions for medical diagnosis [23], [26], [28]-[30]. The resulted knowledge base is often incomplete and inefficient. It may perform poorly in unseen data.

To improve the accuracy of a decision-making system such as an application in medical diagnosis, an integration of symbolic/linguistic processing and numerical computation is motivated to a research area, namely hybrid intelligent architectures [26], [31]-[34]. Hybrid intelligent architectures tend to be more appropriate in applications that require both numerical computation for higher generalization and symbolic/linguistic reasoning for explanation. It is found that hybridization between symbolic/linguistic and numerical representations can achieve higher accuracy compared to either one alone [24], [26], [34].

Most of hybrid intelligent system has focused on the accuracy and the interpretability. In a learning system, an incremental learning capability is considered an important attribute aside from accuracy and interpretability. As for medical diagnosis, patient data grows everyday when new symptoms are discovered. Novel medical knowledge should be quickly incorporated into medical diagnosis system without spending tremendous time in the learning process. In a hybrid system, usually multilayer perceptron (MLP) neural networks are used as a numerical model that is trained by backpropagation algorithms [26]. One well-known problem of the backpropagation is that it is difficult to employ an incremental learning feature. The standard backpropagation algorithm lacks the ability to dynamically generate extra nodes or connections during learning [24], [35]. In medical problems, new knowledge of symptoms may be found after the diagnosis system has been constructed. Using the backpropagation algorithms all old and new data have to be retrained in order to update the knowledge to cover the new symptoms. With an incremental learning algorithm, the new knowledge can be learned and added to the system without retraining previously learned data [35]-[38].

The contribution of this study is to develop a pattern classifier system (i.e., a decision support system) that is concerned not only accuracy and interpretability but also an incremental learning concept. We propose a hybrid intelligent system (HIS) that is composed of numerical model in low level and linguistic model in higher level and is equipped with an incremental learning algorithm. The proposed system is a hierarchical integration of an incremental learning fuzzy neural network (ILFN) [37], [38] and a fuzzy expert system (FES) based on Takagi-Sugeno (TS) fuzzy model [39]. The ILFN is a self-organizing network with the capability of fast online learning. The ILFN can learn incrementally without retraining old information. The linguistic model, FES, is constructed based on knowledge embedded in the trained ILFN. The knowledge captured from the low-level ILFN can be mapped to the higher-level FES and vice versa. The system is equipped with a conflict resolution scheme to maintain consistency in decision-making. The low-level ILFN contributes fast, incremental learning while the higher-level FES offers advantages of dealing with fuzzy data. It provides easy interpretation and explanation to the decision made. The GA [40] is applied to optimize the linguistic model to maintain high accuracy and comprehensibility. The resulted HIS is capable of dealing with low-level numerical data and higher-level linguistic information. After being completely constructed, the system can incrementally learn new information in both numerical and linguistic forms.

The remainder of this paper is organized as follows. Section II discusses the proposed HIS architecture. To demonstrate the effectiveness and efficiency of the proposed system, numerical simulations and benchmark comparisons are presented in Section III. Section IV provides some concluding remarks and future research directions.

II. THE ARCHITECTURE OF THE PROPOSED HYBRID INTELLIGENT SYSTEM

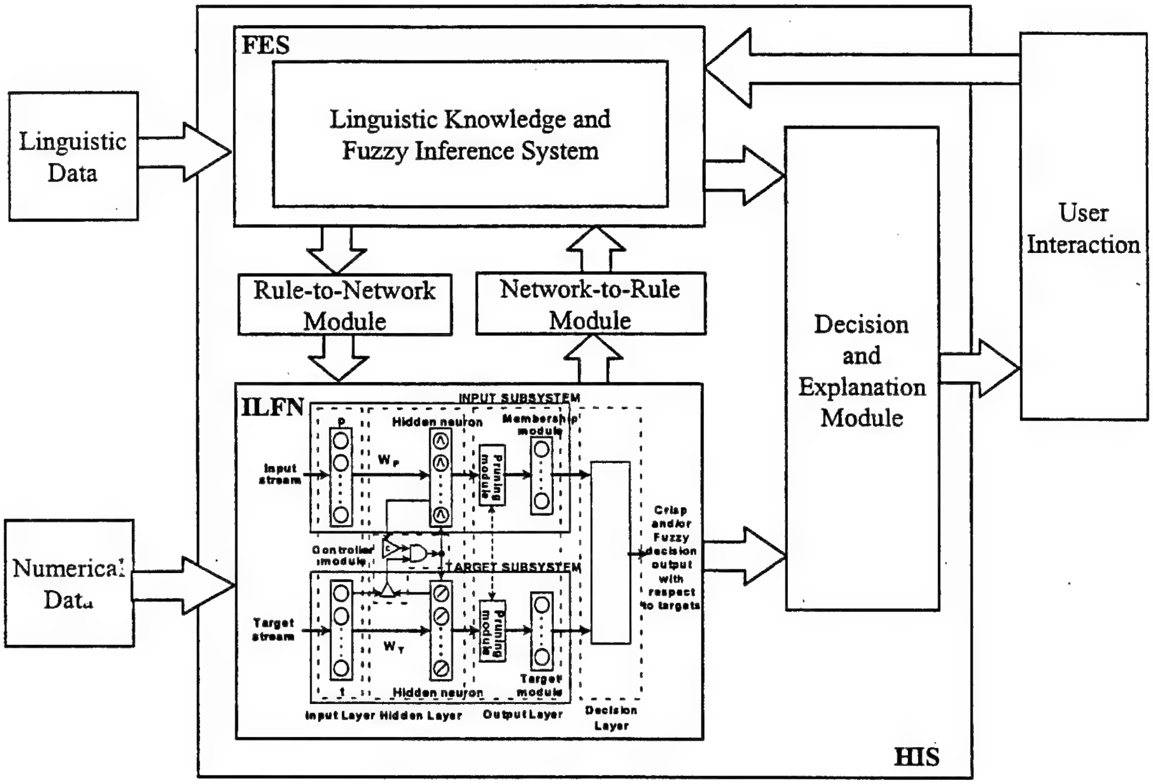


Figure 1: The Architecture of the Proposed Hybrid Intelligent System

The proposed HIS, shown in Figure 1, is constituted of five components: 1) an ILFN, 2) a FES, 3) a network-to-rule module, 4) a rule-to-network module, and 5) a decision-explanation module. Input data is brought into the system through both the ILFN and the FES. The ILFN and the FES are linked together by a network-to-rule module that is a rule extraction algorithm for mapping the ILFN to the FES, and a rule-to-network module that is an algorithm for mapping the FES to the ILFN. The outputs of the ILFN and the FES connect to the decision-explanation module which makes decisions and explanations based on the information received from both the ILFN and the FES. The human operators can interact with the system through a user interaction module. The details of each module are discussed in the following sections.

A. Incremental Learning Fuzzy Neural Network (ILFN)

The ILFN is a self-organizing network that equips with an on-line, incremental learning algorithm capable of learning all training patterns within only one pass. Gaussian radial basis functions are used to form the distribution of the pattern space. The ILFN can learn in a supervised or an unsupervised fashion [37], [38]. The ILFN has four layers: one input layer, one hidden layer, one output layer, and one decision layer, as shown in Figure 2. Alternatively, the system can be viewed as two subsystems: an input subsystem and a target subsystem. Each subsystem has three layers: one input layer, one hidden layer, and one output layer. The hidden layer of both the input subsystem and the target subsystem are linked together via a controller module which is used to control the growing neurons in the hidden layer. Each output layer of both subsystems consists of two modules. The output layer of the input subsystem consists of a pruning module and a membership module, while the output layer of the target subsystem consists of a pruning module and a target module. The membership module of the input subsystem and the target module of the target subsystem are simultaneously updated with their number of neurons controlled by the pruning modules. The output of the classifier is linked together via a decision layer.

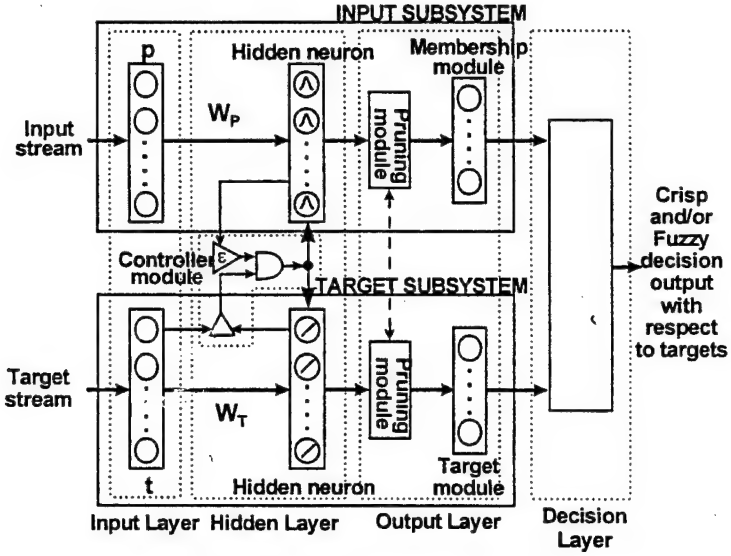


Figure 2: The ILFN Classifier Architecture

ILFN network uses four weighting parameters: $\mathbf{W_P}$, $\mathbf{W_T}$, **STD, count,** as well as one threshold parameter, ε. $\mathbf{W_P}$ is the hidden weight of the input subsystem. Each row (i.e., a node in the hidden layer) of $\mathbf{W_P}$ represents a mean or centroid of a cluster. Each node of $\mathbf{W_T}$ stores the corresponding target of the input prototype patterns. **STD** and **count** are the standard deviation and the number, respectively, of patterns that belong to each node. ε, selected within the range [0, 1], is the threshold parameter that controls the number of clusters. The system generates many clusters if ε is large and few clusters if it is small. However, clusters that belong to the same class are grouped together via the pruning module. The details of learning algorithm can be found in [37], [38].

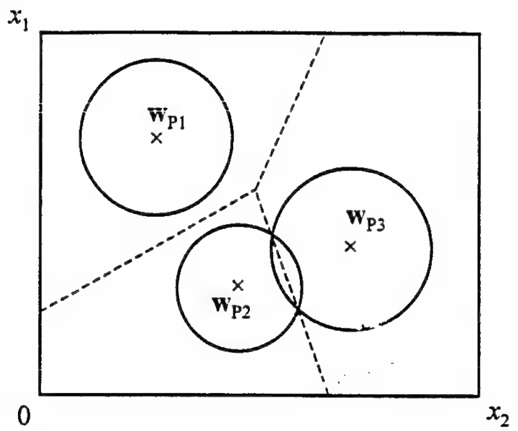


Figure 3: ILFN Decision Boundaries of a Three-Class, Two-Dimensional Pattern Space

Figure 3 shows the ILFN that is used to classify a three-class, two-dimensional pattern space. Three circles are the three clusters of the three classes centered at $\mathbf{w}_{P1}, \mathbf{w}_{P2}$, and $\mathbf{w}_{P3}$. The membership values are highest when the patterns are located at the centers of the clusters. The membership values monotonically decrease when the distances between the patterns and the centers of the clusters increase. The size of the circle depends on the variances of the patterns that belong to the clusters. A pattern outside a circle indicates a near-zero membership degree of belonging to the cluster. The dashed line indicates the boundaries of each cluster.

A trained ILFN does not exhibit a clear meaning of knowledge embedded inside its structure. Linguistic knowledge is more preferable if an explanation about the decision is needed. Thus it is desirable to transform the knowledge of the trained ILFN into a form that is easier to comprehend in linguistic from used in a FES. A FES has a close relationship with an ILFN network in that it can be mapped from one to another. In order to employ both numerical calculation from an ILFN and linguistic processing from a FES, we will combine both a trained ILFN and the mapped FES into the same hybrid system. The output decision of the hybrid system is based on both the ILFN and the FES. The resulting hybrid system would provide complementary features from both the ILFN and the FES. The hybrid system seems to show the ability to deal with more complex problems that need an explanation capability.

The next section describes the details of the FES used in this study, as well as how to map knowledge from a trained ILFN to a FES and vice versa.

B. Fuzzy Expert System (FES)

A FES can be thought of as a special kind of expert systems (ESs). In fact, a FES is an ES that is incorporated with fuzzy sets [41]. Thus, a FES exhibits transparency to users. Users can easily understand the decision made by a FES due to the fact that the rule base is in "if-then" form used in natural languages. From a knowledge representation viewpoint, a fuzzy if-then rule is a scheme for capturing knowledge that is imprecise by nature.

Figure 4 illustrates a schematic diagram of a FES. A FES is composed of four main modules: a fuzzifier, an inference engine, a defuzzifier, and a knowledge base. The function of the fuzzifier is to determine the degree of membership of a crisp input in a fuzzy set. The fuzzy knowledge base is used to represent the fuzzy relationships between input and output fuzzy variables. The output of the fuzzy knowledge base is determined by the degree of membership specified by the fuzzifier. The inference engine utilizes the information from the knowledge base as well as from the fuzzifier to infer additional information. The output of a FES can be fuzzy values from the inference engine process. The output in fuzzy value format is advantage in pattern classification problems since the fuzzy values indicate the degree of belongings of a given pattern to class prototypes. Optionally, the defuzzifier is used to convert the fuzzy output of the system into crisp values.

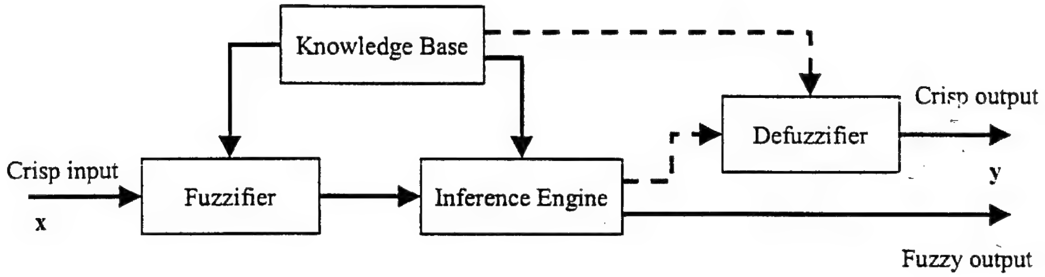


Figure 4: A Fuzzy Expert System (FES)

In a FES, a knowledge base is used for an explanation purpose as well as in a decision making process. A knowledge structure used in the proposed FES comprises of 1) input features' names, 2) variables' ranges, 3) number of linguistic labels, 4) linguistic labels, 5) membership functions, 6) membership functions' parameters, and 7) fuzzy if-then rules. The information about the knowledge structure of the FES can be provided by experts or automatically generated from data. For an *M*-dimensional pattern space, the components in the knowledge base used in the FES are detailed as follows.

Knowledge Base (**K**)

$$\mathbf{K} = \{\ \mathbf{FN, VR, NL, Ling, MF, MP, R}\ \} \tag{1}$$

Input Features' Names (**FN**)

$$\mathbf{FN} = \{fn_1, fn_2, \cdots, fn_M\} \tag{2}$$

Variables' Ranges (**VR**)

$$\mathbf{VR} = \left\{ \begin{array}{cc} Vmin_1, Vmax_1 \\ Vmin_2, Vmax_2 \\ \vdots \quad \vdots \\ Vmin_M, Vmax_M \end{array} \right\} \tag{3}$$

Number of Linguistic Labels (**NL**)

$$\mathbf{NL} = \{N_1, N_2, \cdots, N_M\}; \quad N_j \in \{2, \cdots, 9\} \tag{4}$$

To maintain comprehensibility of the linguistic model, the number of the linguistic variables should be as small as possible. It is suggested that it should not be larger than nine [42].

Linguistic Labels (**Ling**)

$$\mathbf{Ling} = \begin{Bmatrix} \{l_{11}, l_{12}, \cdots, l_{1N_1}\} \\ \{l_{21}, l_{22}, \cdots, l_{2N_2}\} \\ \vdots \\ \{l_{M1}, l_{M2}, \cdots, l_{MN_M}\} \end{Bmatrix} \quad (5)$$

where l_{jk} is a linguistic label in the j^{th} dimension and k is the index to it.

Membership Functions (**MF**)

$$\mathbf{MF} = \begin{Bmatrix} \{mf_{11}, mf_{12}, \cdots, mf_{1N_1}\} \\ \{mf_{21}, mf_{22}, \cdots, mf_{2N_2}\} \\ \vdots \\ \{mf_{M1}, mf_{M2}, \cdots, mf_{MN_M}\} \end{Bmatrix} \quad (6)$$

Membership Functions' Parameters (**MP**)

$$\mathbf{MP} = \begin{Bmatrix} \{\mathbf{mp}_{11}, \mathbf{mp}_{12}, \cdots, \mathbf{mp}_{1N_1}\} \\ \{\mathbf{mp}_{21}, \mathbf{mp}_{22}, \cdots, \mathbf{mp}_{2N_2}\} \\ \vdots \\ \{\mathbf{mp}_{M1}, \mathbf{mp}_{M2}, \cdots, \mathbf{mp}_{MN_M}\} \end{Bmatrix} \quad (7)$$

$$\mathbf{mp}_{jk} = \{ mp_{jk1}, mp_{jk2}, \cdots, mp_{jkn_{jk}} \};\ j = 1, \ldots, M;\ k = 1, \ldots, N_j. \quad (8)$$

Assume that mf_{jk} is a Gaussian membership function, $\mathbf{mp}_{jk}$ has two variables which are σ and μ, a standard deviation and a mean of a Gaussian membership function, respectively. Thus, we have

$$\mathbf{mp}_{jk} = \{mp_{jk1}, mp_{jk2}\} = \{\sigma_{jk}, \mu_{jk}\};\ j = 1, \ldots, M;\ k = 1, \ldots, N_j. \quad (9)$$

Fuzzy If-Then Rules (**R**)

$$\mathbf{R} = \{\mathbf{A}_{L\times M}, \mathbf{B}_{L\times 1}, \mathbf{CF}_{L\times 1}\} = \begin{Bmatrix} A_{11} & A_{12} & \cdots & A_{1M} & B_1 & CF_1 \\ A_{21} & A_{22} & \cdots & A_{2M} & B_2 & CF_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ A_{L1} & A_{L2} & \cdots & A_{LM} & B_L & CF_L \end{Bmatrix}. \quad (10)$$

$\mathbf{A}_{L\times M}$ represents the antecedent part of the if-then rules; $\mathbf{B}_{L\times 1}$ and $\mathbf{CF}_{L\times 1}$ constitute the consequent part of the if-then rules; where L and M is the number of fuzzy rules and the dimension of the pattern space, respectively. A_{ij}, $i = 1, \ldots, L$, $j = 1, \ldots, M$, is the antecedent of the i^{th} rule for the j^{th} dimension. $A_{ij} \in \{0, 1, \cdots, N_j\}$ is the index of a linguistic label in the j^{th} dimension of the linguistic labels (**Ling**) of the i^{th} rule. If A_{ij} is "0" then the system uses a *don't care* label in which its activation function is always a unity membership grade. B_i is a constant value that is a class consequent part of the i^{th} rule. CF_i, a value in [0, 1], is a confident factor of the i^{th} rule. In a FES, for a finite class pattern classification problem with an M-dimensional pattern space, linguistic knowledge can be written as a set of fuzzy if-then rule in a natural language as follow:

R_i: IF x_1 is A_{i1} AND x_2 is A_{i2} AND … AND x_M is A_{iM},
THEN $\mathbf{x} = \{x_1, x_2, \ldots, x_M\}$belongs to Class B_i with confident factor = CF_i; (11)

where R_i, i = 1, …, L, is the label of the i^{th} rule and A_{ij} indicates a linguistic label such as *small*, *medium*, or *large*.

Assume that Gaussian membership functions are employed in the FES. The rule firing strength ϕ_i can be computed by the following equation:

$$\phi_i = \min_j \{ \exp\left[-\left(\frac{x_j - \mu_{ij}}{\sigma_{ij}} \right)^2 \right] \}, \text{ for } i = 1, \ldots, L; j = 1, \ldots, M; \tag{12}$$

where μ_{ij} and σ_{ij} are the mean and the standard deviation, respectively, of the linguistic label indexed by A_{ij}; and *min* is a T-norm operator which can be replaced by *product*.

After computing the firing strength from each rule, the class output, C_y, is calculated by using the inference mechanism as follow:

$$C_y = B_J ; J = \arg \max_i (\phi_i \times CF_i). \tag{13}$$

In developing a FES, developers must pay attention to several issues such as accuracy, comprehensibility, compactness, completeness, and consistency. *Accuracy* is a quantitative measure that indicates the performance of a FES in classifying both training and testing data. *Comprehensibility* indicates how easily a FES can be accessible by human beings. Generally, comprehensibility of a FES depends on the following aspects: the distinguishability of the shapes of membership functions, the number of fuzzy if-then rules, and the number of antecedent conditions of fuzzy if-then rules. *Compactness* involves the size of fuzzy if-then rules and the number of antecedents of fuzzy if-then rules. More compactness of a fuzzy system usually yields higher comprehensibility. *Completeness* assures that a FES will provide a non-zero output for any given input in the input space. *Consistency* makes sure that fuzzy if-then rules are not conflicting with each other as well as human senses. Fuzzy if-then rules are inconsistent if they have very similar antecedents, but different consequents, and they conflict with the expert knowledge. If there are fuzzy if-then rules that are conflicting to each other, the rules become unclear [43]-[46]. The conflicting rules need to be resolved.

For the completeness of the rule structure in a FES, grid partition methods are widely used for partitioning input space into grid cells. Fuzzy if-then rules can be obtained by using fuzzy grid partitions [47]. Despite its advantage of providing the completeness of rule structure, the grid partition method has a disadvantage in that the number of fuzzy rules increases exponentially as the dimension of the input space increases. Since each cell represents a fuzzy if-then rule, the number of fuzzy if-then rules is usually very large. The system becomes a black-box scheme that is not comprehensible to human users. Pattern classification problems in real world often have large dimensions. It is undesirable to directly use the grid-type partitioning for constructing fuzzy if-then rules.

To obtain a smaller number of fuzzy rules, projection from clusters [43], [45] is called for. Clustering algorithms can be used for partitioning data points into a small number of clusters. Each cluster then represents a fuzzy relation and corresponds to a rule. The fuzzy sets in the antecedent parts of the rules are projected from the clusters onto the corresponding axis of the data space. The number of rules from projection method is smaller than grid-type partitioning. A more compact linguistic model is obtained. However, the fuzzy sets that are directly projected from clustering methods may not be transparent enough. The number of fuzzy sets from the projection may be very large and redundant since the projected fuzzy sets may be very similar resulting in a fuzzy system that is not optimal. The problem mentioned above can be solved using rule simplification methods [45], [48]. Alternatively, the projection from trained ILFN parameters onto fuzzy if-then rules.

The ILFN groups the patterns in the input space into a small number of clusters. Based on grid partition methods, the clusters and its parameters of the trained ILFN can be mapped to fuzzy if-then rules. The number of fuzzy if-then rules is equal to the number of clusters in the trained ILFN. The number of fuzzy sets in each dimension depends on the number of grid partitions chosen. The parameters of fuzzy sets are projected from the cluster parameters of the trained ILFN. Figure 5a shows the projection of ILFN to one-dimensional fuzzy sets. We can see that the fuzzy sets in Figure 5a have some similarity. Combining grid partitioning method and projection method is shown in Figure 5b. The parameters of fuzzy sets in Figure 5b projected and adapted from the clusters of the trained ILFN.

Using a grid-based projection method, the fuzzy if-then rules of the FES are extracted from a trained ILFN. The hidden numerical weights of the ILFN are mapped into initial fuzzy if-then rules. A genetic algorithm is then used to select only discriminatory features resulting in a more compact rule set with highly transparent linguistic terms. The following section describes the mechanism used to map the ILFN to the FES.

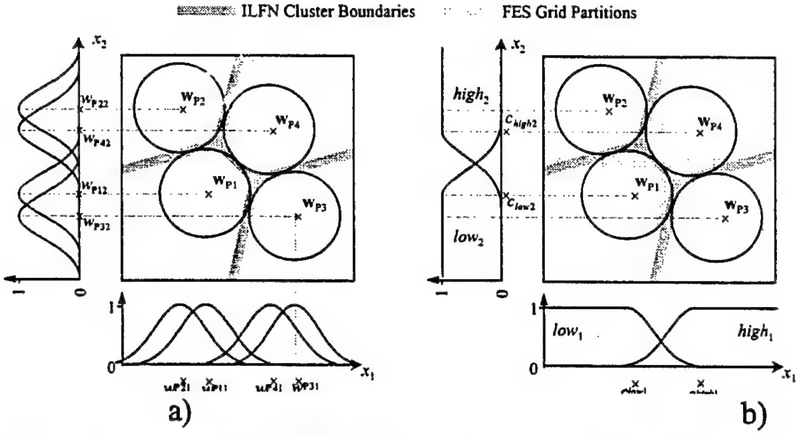


Figure 5: a) Projection of ILFN to One-Dimensional Fuzzy Sets
b) FES Grid Partition with its Parameter Projected from Trained ILFN

C. Network-To-Rule Module

Since the knowledge embedded in the ILFN is not in linguistic form, the ILFN lacks of an explanation capability. ILFN weights can be extracted by using a rule extraction algorithm to obtain linguistic rules. A meaningful explanation in reasoning process can then be generated from the linguistic model i.e., the FES. The mechanism used for mapping a trained ILFN to a linguistic knowledge base operates inside the network-to-rule module. The mechanism is called *ilfn2rule* algorithm.

1) ILFN2RULE Algorithm

Using a grid-based projection method, the ilfn2rule algorithm is used to map a trained ILFN to fuzzy if-then linguistic rules.

The user specifies membership functions' types (**MF**). Any type of fuzzy membership functions can be used. In this study, Gaussian membership functions are used. The rule extraction algorithm is given in four steps described below.

Step 1: Retrieve trained ILFN parameters (**W**$_P$, **W**$_T$, **count**) as well as the numbers of linguistic labels (**NL**). (The numbers of linguistic labels are determined during the genetic optimization process that will be discussed later.)

Step 2: Calculate membership functions' parameters (**MP**) that are a center and a standard deviation for each linguistic label in the case that Gaussian membership function is used. Centers of Gaussian functions can be determined from the variables' ranges (**VR**) that are minimum and maximum values of the numerical weight **W**$_P$ for the ILFN network.

$$Vmin_j \;=\; \min\,(w_{P1j}, w_{P2j}, \cdots, w_{PLj}) \;=\; \min_i\,(w_{Pij}), \tag{14}$$

$$Vmax_j \;=\; \max\,(w_{P1j}, w_{P2j}, \cdots, w_{PLj}) \;=\; \max_i\,(w_{Pij}), \tag{15}$$

$$res_j \;=\; \frac{Vmax_j - Vmin_j}{N_j \text{ -}1}, \tag{16}$$

where i = 1, …, L; j = 1, …, M; L is the number of hidden nodes, i.e., prototypes created by the ILFN network; M is the dimension of the pattern space; res_j represents the numerical resolution between linguistic variables in the j^{th} dimension; $Vmax_j$ and $Vmin_j$ are the maximum and the minimum values of the weight **W**$_P$ in the j^{th} dimension; and N_j is the number of linguistic variables in the j^{th} dimension.

$$\mu_{jk} = \begin{cases} Vmin_j & \text{for } k = 1 \\ \mu_{j,k-1} + res_j & \text{for } k = 2, \ldots, N_j \end{cases}, \tag{17}$$

$$\mu_j = [\,\mu_{j1}, \mu_{j2}, \ldots, \mu_{jN_j}\,], \tag{18}$$

$$\sigma_j \;=\; \sqrt{\text{-}\left(\frac{res_j^2}{\sqrt{2} \times \ln\lambda}\right)}, \tag{19}$$

where μ_{jk}, k = 1, …, N_j, represents the mean of the k^{th} Gaussian membership function in the j^{th} dimension; σ_j represents the standard deviation of the Gaussian membership functions in the j^{th} dimension; and λ, selected in [0, 1], represents the overlap parameter between membership functions.

Step 3: Map the numerical weight **W**$_P$ into linguistic label form using the following equation:

$$A_{ij} \;=\; \arg\min_k\big(dist(w_{Pij}, cen_{jk})\big), \tag{20}$$

where A_{ij} represents the index of the linguistic label mapped from w_{Pij}; and w_{Pij}, for $i = 1, \ldots, L, j = 1, \ldots, M, k = 1, \ldots, N_j$, is an element of the hidden weight $\mathbf{W_P}$ of the ILFN network. M is the dimension of the pattern space and L is the number of prototypes created by the ILFN network.

Step 4: Generate if-then rule table: use linguistic antecedent parts obtained from $\mathbf{W_P}$ and consequent parts from $\mathbf{W_T}$. The number of fuzzy if-then rules is equal to the number of hidden neurons of the trained ILFN. Calculate confident factor CF_i, $i = 1, \ldots, L$, for each rule using count parameter **count** using the following equation:

$$\mathbf{count} = [cnt_1, cnt_2, \ldots, cnt_L]^{\mathrm{T}}, \tag{21}$$

$$CF_i = \frac{cnt_i}{\sum\limits_{h \in Class(w_{Ti})} cnt_h}, \tag{22}$$

where cnt_i, $i = 1, \ldots, L$, is a count parameter of the i^{th} rule (i.e., the i^{th} prototype) obtained when a pattern is included into the i^{th} prototype; and $Class(w_{Ti}) = \{l \mid w_{Tl} = w_{Ti}, l = 1, \ldots, L\}$.

$$\mathbf{CF} = [CF_1, CF_2, \ldots, CF_L]^{\mathrm{T}} \tag{23}$$

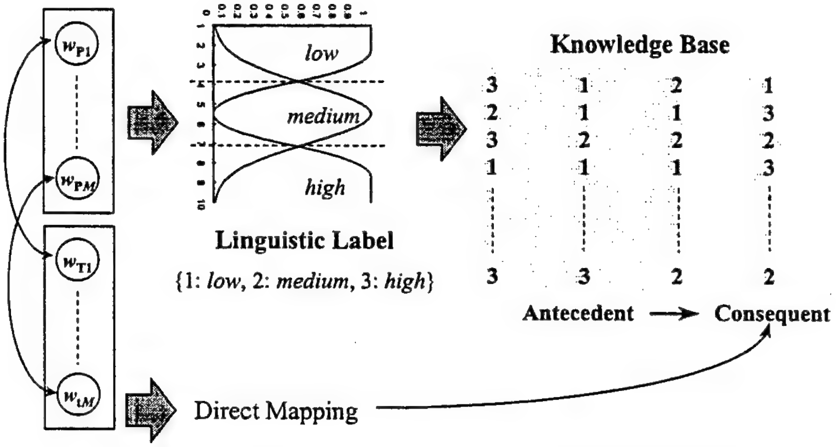


Figure 6: Mapping from ILFN to Linguistic Rules

The knowledge base from Figure 6 can be described by fuzzy linguistic form that is similar to natural language as follows:

Rule 1: If feature$_1$ is *high* and feature$_2$ is *low* and feature$_3$ is *medium*, then class is 1;
Rule 2: If feature$_1$ is *medium* and feature$_2$ is *low* and feature$_3$ is *low*, then class is 3;
Rule 3: If feature$_1$ is *high* and feature$_2$ is *medium* and feature$_3$ is *medium*, then class is 2;
...

After linguistic rules are extracted from the ILFN network, they can be used as a rule base for a fuzzy expert system. A fuzzy expert system is considered as a higher-level knowledge representation since it uses if-then rules similar to natural languages. Using linguistic form makes the system transparent allowing human users to easily comprehend the rationale of how the decision was made. Explanations and answers can be provided if needed.

In pattern classification problems, the dimension of the pattern space may be very large. For very large dimensions, it is too cumbersome to use all the features available as a knowledge base. Though it is described in linguistic form, using all available features, it results in a system that is no longer transparent to users. It is possible to select only feature subset that provides the most discriminatory power in classifying patterns. To do this, the genetic algorithm is very useful and suitable to select the important features. We will adapt the genetic algorithm (GA) [40] to search for an optimal number of features used for each rule while maintaining a high percentage of correct classification. This will result in reducing the number of rules as well. Some rules will be redundant after many features have been eliminated, these duplicate rules can be pruned out.

2) Genetic Algorithm for Rule Optimization

The linguistic rule base extracted from the ILFN is sub-optimal. In order to obtain a near optimal rule set, the GA is used to operate on initial fuzzy rules. An integer chromosome representation is used instead of a binary chromosomes representation, to reduce the size of chromosome and improve the speed of the evolutionary operations. The fuzzy if-then rules are encoded into integer chromosomes to be evolved by the GA. After converging, the best chromosomes are decoded back into the FES with a compact rule set.

In order to apply the genetic optimization, the if-then rule base is encoded in a chromosome representation. Only the antecedent is coded and operated on by the evolutionary process. The original rule set is used as a reference rule set in decoding the final population to the final linguistic rule base.

3) Fuzzy If-Then Rule Encoding

In our procedure, only the antecedents of the if-then rules are used in genetic encoding. If-then rules are encoded to an integer chromosome. A chromosome sometimes refers to an individual of the population. The elements of each chromosome are called genes that are integer numbers. Each gene in a chromosome can be decoded to a fuzzy if-then rule. Let $\mathbf{G}_R$ be a chromosome that is a set of genes g_i, $i = 1,\ldots, L$, where

$$\mathbf{G}_R = \{ g_1, g_2, \ldots, g_L \} \tag{24}$$

$$g_i = \sum_{j=1}^{M} 2^{j-1} \times a_{ij} \tag{25}$$

$$a_{ij} = \begin{cases} 0 & \text{if } A_{ij} = 0 \\ 1 & \text{otherwise} \end{cases} \; ; i = 1,\ldots,L, j = 1,\ldots, M, \tag{26}$$

where L is the number of fuzzy if-then rules; M is the dimension of the pattern space. The antecedent A_{ij} is the linguistic label in the jth dimension of the ith rules. a_{ij} is equal "1" meaning that the jth dimension of the ith rule is being used and a_{ij} is equal "0" meaning that the jth dimension of the ith rule is not being used, i.e., *don't care*.

For example, a FES is used in a three-class, two-dimensional problem space. The fuzzy expert system has two linguistic labels {1: *low*, 2: *high*} in each dimension. Suppose that there are four rules extracted from a trained ILFN, as follows.

Rule 1: if x_1 is *low* and x_2 is *high*, then class 1;
Rule 2: if x_1 is *high* and x_2 is *high*, then class 2;
Rule 3: if x_1 is *low* and x_2 is *low*, then class 3;

Rule 4: if x_1 is *high* and x_2 is *low*, then class 3.
That is we have **R** = { **A**, **B** },

$$\mathbf{A} = \left\{\begin{matrix} 1 & 2 \\ 2 & 2 \\ 1 & 1 \\ 2 & 1 \end{matrix}\right\},\ \mathbf{B} = \left\{\begin{matrix} 1 \\ 2 \\ 3 \\ 3 \end{matrix}\right\}$$; where **A** is the antecedent set and **B** is the consequent set.

Let $\underline{\mathbf{A}}$ be a set of antecedents when some features are composed of a *don't care* linguistic label. Let $\underline{a}$ be a binary set of 1's and 0's indicating whether or not an element of **A** is used.

Suppose the antecedent set **A** is reduced to $\underline{\mathbf{A}} = \left\{\begin{matrix} 1 & 2 \\ 2 & 2 \\ 0 & 0 \\ 0 & 1 \end{matrix}\right\}$.

That is we have a binary set $\underline{\mathbf{a}} = \left\{\begin{matrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 1 \end{matrix}\right\}$.

From the antecedent set $\underline{\mathbf{A}}$, we have an encoded chromosome

$$\begin{aligned} \mathbf{G}_{\mathrm{R}} &= \{(2^{1-1}\times 1+2^{2-1}\times 1), (2^{1-1}\times 1+2^{2-1}\times 1), (2^{1-1}\times 0+2^{2-1}\times 0), (2^{1-1}\times 0+2^{2-1}\times 1)\} \\ &= \{ (1+2), (1+2), (0+0), (0+2) \} \\ &= \{ 3, 3, 0, 2 \}. \end{aligned}$$

4) Fuzzy If-Then Rule Decoding

Integer chromosomes are used in the genetic optimization process. After convergence of the solution, encoded integer chromosomes are decoded back to fuzzy if-then rule bases. Given an integer chromosome, each gene is decomposed into binary format. The decoding process is an inverse process of the encoding process mentioned above. For example, suppose that we have a solution chromosome $\mathbf{G}_{\mathrm{R}}$ = { 3, 3, 0, 2}. $\mathbf{G}_{\mathrm{R}}$ can be decomposed to binary format as follows:

$$\begin{aligned} \mathbf{G}_{\mathrm{R}} &= \{ 3, 3, 0, 2 \} \\ &= \{ (1+2), (1+2), (0+0), (0+2) \} \\ &= \{(2^{1-1}\times 1+2^{2-1}\times 1), (2^{1-1}\times 1+2^{2-1}\times 1), (2^{1-1}\times 0+2^{2-1}\times 0), (2^{1-1}\times 0+2^{2-1}\times 1)\}. \end{aligned}$$

So, we have $\underline{\mathbf{a}} = \left\{\begin{matrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 0 & 1 \end{matrix}\right\}$. Knowing the origin antecedent set $\mathbf{A} = \left\{\begin{matrix} 1 & 2 \\ 2 & 2 \\ 1 & 1 \\ 2 & 1 \end{matrix}\right\}$, we have the reduced antecedent $\underline{\mathbf{A}} = \left\{\begin{matrix} 1 & 2 \\ 2 & 2 \\ 0 & 0 \\ 0 & 1 \end{matrix}\right\}$; $\quad \underline{\mathbf{R}} = \{ \underline{\mathbf{A}}, \mathbf{B} \} = \left\{\begin{matrix} 1 & 2 & 1 \\ 2 & 2 & 2 \\ 0 & 0 & 3 \\ 0 & 1 & 3 \end{matrix}\right\} = \left\{\begin{matrix} 1 & 2 & 1 \\ 2 & 2 & 2 \\ 0 & 1 & 3 \end{matrix}\right\}$.

Note that if a rule comprises of all *don't care* linguistic labels in the antecedent part, then that rule can be eliminated.

5) Genetic Selection for the Number of Linguistic Variables

The number of linguistic variables can be varied depending on a given problem. Some problems may need more linguistic variables than others. Using more linguistic variables results in finer fuzzy partitions and better classification performance. However, to maintain the interpretability of the system, the number of linguistic variable should be as small as possible. Selecting of the numbers of linguistic variables becomes a trade off between the accuracy of the system and the interpretability of the system. To obtain the optimal point that balances between the accuracy and the interpretability is not an easy task in selecting the number of linguistic variables. To avoid the difficulty, the numbers of linguistic labels can be selected by using the genetic algorithm. The genetic selection for the linguistic numbers can be processed simultaneously with the rule optimization.

The chromosome for the genetic optimization of the number of linguistic variables ($\mathbf{G}_{\mathrm{NL}}$) can be written as

$$\mathbf{G}_{\mathrm{NL}} = \{ N_1, N_2, \ldots, N_M \} \tag{27}$$

where N_j, $j = 1, \ldots, M$, is the number of linguistic variables for the j^{th} dimension. The chromosome for optimizing the number of linguistic variables ($\mathbf{G}_{\mathrm{NL}}$) can be combined with the chromosome for optimizing the fuzzy if-then rules ($\mathbf{G}_{\mathrm{R}}$). The combined chromosome from equations (24) and (27) can be written as

$$\mathbf{G} = \{\mathbf{G}_{\mathrm{NL}}, \mathbf{G}_{\mathrm{R}}\} = \{ N_1, N_2, \ldots, N_M, g_1, g_2, \ldots, g_L \} \tag{28}$$

6) The Genetic Algorithm

When the genetic algorithm is implemented, it usually proceeds in a manner that involves the following steps:

Step 1 Initialization of the population
Step 2 Fitness evaluation
Step 3 Mate selection
Step 4 Crossover
Step 5 Mutation
Step 6 Check stopping criteria; if the solution meets the criteria, stop the algorithm and obtain the final if-then rules; otherwise, repeat Steps 2-6.

Initialization of the population: A chromosome has two different groups of genes: the number of linguistic variables and the fuzzy if-then rules. The initial population of the chromosomes is randomly selected as integer numbers in both of the groups. These initial individuals will be reproduced to next generation via the genetic operations: fitness evaluation, mate selection, crossover, and mutation.

Fitness evaluation: The fitness function is based on the performance of resulting rules decoded from a chromosome and the compactness of the rule set. A fuzzy expert system with the decoded rules is used to evaluate the performance of the resulting rules. The fitness function of a chromosome **G** can be determined from the following equations:

$$fitness(\mathbf{G}) = W_{\mathrm{PC}} \times PC - W_{\mathrm{F}} \times SC - W_{\mathrm{NL}} \times NL, \tag{29}$$

$$PC = \frac{Total\ Patterns - Wrongly\ Classified\ Patterns}{Total\ Patterns} \times 100, \tag{30}$$

$$SC = \sum_{i,j} a_{ij}, \tag{31}$$

$$NL = \sum_{j} N_j \tag{32}$$

where W_{PC} is the weight of percent correct classification by a fuzzy expert system; PC is the percent correct classification; W_{F} is the weight of the number of features used for a rule set; a_{ij} is calculated from (26); SC is the structure complexity of the fuzzy system i.e., number of features used for a rule set, i.e., number of 1's in $\underline{\mathbf{a}}$; W_{NL} is the weight of the number of linguistic variables used for a rule set; and NL is the summation of the numbers of linguistic variables used. Preferring fewer linguistic variables, fewer rules, and fewer features with higher correct classification performance, the weight of percent correctly classified patterns (W_{PC}), is usually set to be relatively larger than the weight of structure complexity (W_{F}) and the weight of the linguistic variables (W_{NL}). W_{F}, W_{PC}, and W_{NL} are all positive numbers in $\Re$; they are predefined by the user.

Mate selection: There are many ways of selecting individuals for mating. One of the well-known methods is *roulette wheel selection* [49]. The fittest individuals usually have a higher chance to mate than the ill-fitted ones. In roulette wheel selection, the individuals are randomly selected based on the probability of fitness. The reproduction probability can be defined from the fitness function.

$$prob(\mathbf{G}_i) = \frac{fitness(\mathbf{G}_i)}{\sum_{j=1}^{P} fitness(\mathbf{G}_j)}, \quad i = 1, \ldots, P, \tag{33}$$

where P is the number of individuals in the population.

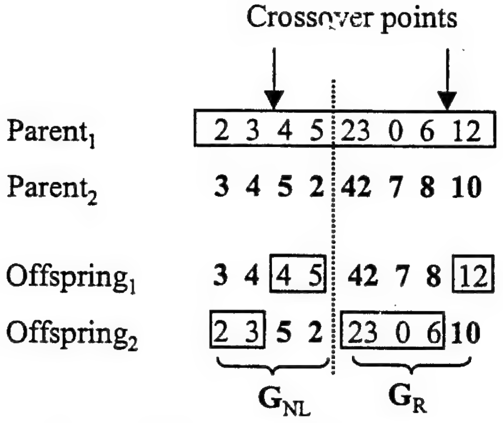


Figure 7: Crossover Operation

Crossover: After mate selection operation, crossover operation is performed. Crossover operation is a mechanism for changing information between two chromosomes called *parents* to reproduce two new individuals called *offspring*. A crossover point is selected randomly with probability p_c. In our problem, since a chromosome is separated into two groups, the crossover process is also separated into two parts: the crossover of the number linguistic variables and the crossover of fuzzy if-then rules. The crossovers of the two parts are independent from each other. Figure 7 illustrates how two chromosomes crossover, yielding two offspring.

Mutation: Mutation is applied to offspring to prevent the solution from trapping at a local minimum area. The mutation operation allows the genetic algorithm to explore new possible solutions and increase a chance to get near global minima. Figure 8 illustrates how the mutation operation works.

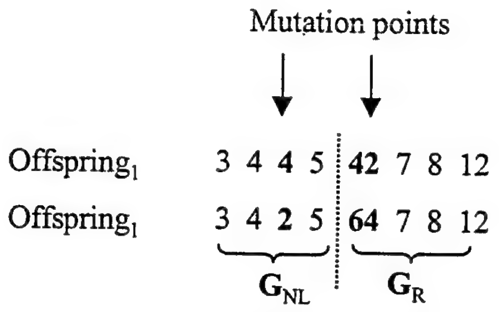


Figure 8: Mutation Operation

An offspring chromosome mutates with the mutation probability p_m on each gene. In the integer-coded genetic algorithm, the mutation process operates from the following equation

$$G_{new} = \text{round}(G_{old} + \gamma \times \text{randn}(1)), \tag{34}$$

where G_{old} is a gene selected for mutating; G_{new} is the resulted gene from mutating; *randn*(1) is a random number in [0, 1] produced by the Gaussian random generator; and γ is the highest possible integer value a gene is allowed to be. The two parts of the chromosome **G** have different values. The highest possible value of the number of linguistic variables is set to 9 or smaller. The highest integer value for the gene of the if-then rule is 2^M for M-dimensional space.

D. Rule-To-Network Module

The rule-to-network module is used for transferring the linguistic knowledge into the ILFN structure. The rule-to-network module allows an expert to incorporate his knowledge into the system. The rule-to-network consist of the *rule2ilfn* algorithm that is used for mapping the FES to the ILFN.

1) RULE2IIFN Algorithm

There are two phases in the rule2ilfn algorithm. Phase 1 is used when fuzzy rules are compact where they have *don't care* linguistic variables. The *don't care* linguistic variables need to be transformed into intermediate rules. In the transformed intermediate rules, every feature or component of the rules is composed of at least a linguistic variable attached; otherwise, we cannot map rules to an ILFN network. Phase 2 operates after phase 1 ended. In phase 2, the parameters of fuzzy rules are correspondingly mapped to the parameters of an ILFN network. The details of the two phases are as follows.

```
Phase 1: Mapping a compact rule set to an intermediate rule set.
    1) Retrieve a rule $R_i = \{A_{i1}, A_{i2},\ldots, A_{iM}, B_i, CF_i\}$; $i = 1, \ldots, L$.
    2) Check for a feature that has a don't care linguistic label (i.e., $A_{ij} = 0$). Within
       the present rule, if there is a feature having a don't care linguistic label;
       expand every possible rule to cover the combinations of available linguistic
       labels.
    3) Repeat 1) and 2), until there are no more rules.
    4) Output the intermediate rule set.
Phase 2: Mapping an intermediate rule set to an initial ILFN network.
    ▪ Set $\mathbf{W}_P$, $\mathbf{W}_T$, STD, and count to be empty sets.
    ▪ For $i^{th}$ rule = 1 to $L$ do,
        • For $j^{th}$ feature = 1 to $M$ do,
            o Set $w_{Pij} = \mu_{ij}$
            o Set $std_{ij} = \sigma_{ij}$
        • Set $w_{Ti} = B_i$
        • Set $cnt_i = 1$
    ▪ $\mathbf{W}_P = [\mathbf{w}_{P1}, \mathbf{w}_{P2}, \ldots, \mathbf{w}_{PL}]^T$; where $\mathbf{w}_{Pi} = [w_{Pi1}, w_{Pi2},\ldots, w_{PiM}]$
    ▪ $\mathbf{W}_T = [w_{T1}, w_{T2}, \ldots, w_{TL}]^T$
    ▪ $\mathbf{STD} = [\sigma_1, \sigma_2, \ldots, \sigma_L]^T$; where $\sigma_i = [\sigma_{i1}, \sigma_{i2},\ldots, \sigma_{iM}]$
    ▪ $\mathbf{count} = [cnt_1, cnt_2,\ldots,cnt_L]^T$
```

L is the number of rules and M is the dimension of pattern space. The parameters μ_{ij} and σ_{ij}, $i = 1, \ldots, L$, $j = 1, \ldots, M$, are a mean and a standard deviation of the linguistic label indexed by A_{ij}. More specifically, μ_{ij} and σ_{ij} are taken from $\mathbf{mp}_{jA_{ij}}$ that is in the membership functions' parameters (**MP**) from Equation (7).

After obtaining the initial ILFN network, available training data is used to refine the ILFN network. Network pruning is also needed to eliminate the hidden nodes that do not have any belonging pattern. This can be done by checking at the parameter **count**. If count of a node is equal to one, then eliminate that node.

E. The Decision-Explanation Module

The last module in the HIS is the decision-explanation module. The decision-explanation module performs two functions: making a decision and explaining the decision. For a first function, making a decision, the decision-explanation module receives two inputs from the outputs of low-level ILFN and higher-level FES. Another function of the decision-explanation module is to generate a natural language to explain and conclude the decision made by using the knowledge base (**K**) from the FES module.

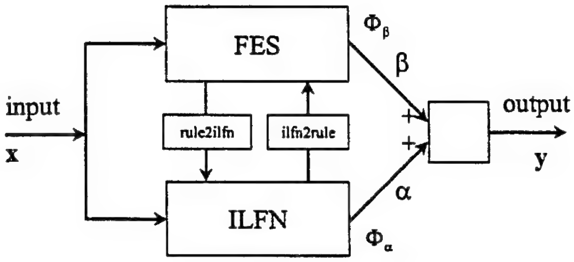


Figure 9: Hybrid System Combined from ILFN and FES

Figure 9 shows an equivalent of the HIS. The decision for the class output C_y can be calculated by the following equations:

$$\mathbf{C} = [C_1, C_2, \cdots, C_Q] \tag{35}$$

$$\Phi_\alpha = [\phi_{\alpha 1}, \phi_{\alpha 2}, \cdots, \phi_{\alpha Q}] \tag{36}$$

$$\Phi_\beta = [\phi_{\beta 1} \times CF_1, \phi_{\beta 2} \times CF_2, \cdots, \phi_{\beta Q} \times CF_Q] \tag{37}$$

$$y_i = \frac{\alpha_i \phi_{\alpha i} + \beta_i \phi_{\beta i} \times CF_i}{\alpha_i + \beta_i}, \text{ for } i = 1, \ldots, Q \tag{38}$$

$$\mathbf{y} = [y_1, y_2, \cdots, y_Q] \tag{39}$$

$$C_y = C_J ; J = \arg\max_i (y_i) \tag{40}$$

where **C** is the class vector; Φ_α is the membership values from the ILFN with respect to **C**; Φ_β is the membership values from the FES with respect to **C**; **y** is the membership values from the HIS with respect to **C**; $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_Q]$ and $\beta = [\beta_1, \beta_2, \ldots, \beta_Q]$ are the real-value weights linking from the ILFN and the FES to the decision-explanation module, respectively; Q is the number of classes; and C_y is the class decision output from the HIS. Please note that α and β can be specified by used or determined by an optimization algorithm such as the GA. In our study we used a real GA to search for possibly optimal values of α and β.

For simplicity, we may set $\alpha = \alpha_1 = \alpha_2 = \ldots = \alpha_Q$ and $\beta = \beta_1 = \beta_2 = \ldots = \beta_Q$. Then we have

$$\mathbf{y} = \frac{\alpha \Phi_A + \beta \Phi_B}{\alpha + \beta} \tag{41}$$

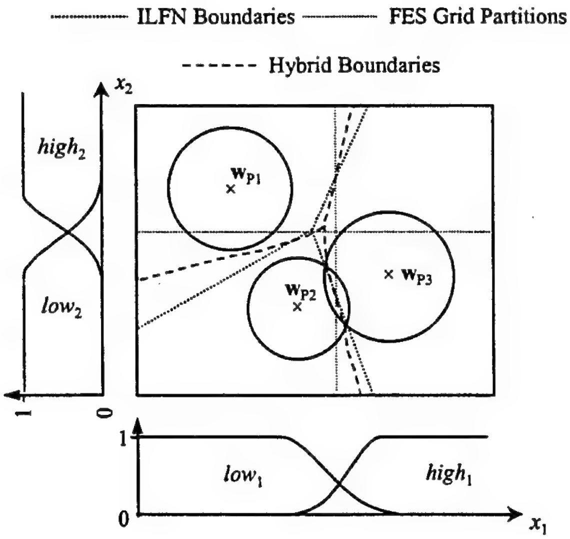


Figure 10: Hybrid Decision Boundaries

1) Decision Boundaries

The decision boundaries of the HIS come from the weighted average of the boundaries from the ILFN and FES. The decision boundaries of the ILFN and the decision boundaries of the FES contribute in different manners. The decision boundaries of the ILFN emphasize in local area to achieve better generalization while the decision boundaries of the FES preserve for human interpretability. Since numerical information in the ILFN and linguistic information in the FES have complement benefits, it is preferable to incorporate both structures into the same system. The boundaries of the hybrid system provide both accuracy and interpretability. In the HIS, the ILFN serves as a low-level numerical computation, while the FES operates as a higher-level linguistic computation. Hybrid weights (α and β) play important role in adjusting the hybrid decision boundaries. If α_i is larger than β_i, the hybrid boundaries tend toward the ILFN boundaries. If α_i is smaller than β_i, the hybrid boundaries tend toward the FES boundaries. Figure 10 shows the hybrid decision boundaries of the combined ILFN and FES.

2) Conflict and Conflict Resolution Between Low Level and Higher Level

Ideally, there should be no conflict between low level and higher level decisions in the HIS, if it is a one-to-one mapping between them. Since a combination of grid based partition and projection is used in the mapping process, decisions from ILFN and FES may conflict with each other. The diagram in Figure 11 shows the possible conflict decisions between the ILFN and the FES. The system is conflict, if the decision from the ILFN is correct but the decision from the FES is wrong or if the decision from the ILFN is wrong but the decision from the FES is correct. The system is not conflict, if the two systems make the same decision. It is preferable that the two systems are not conflict and both make correct decisions.

It is feasible to resolve the conflict between the two systems by forcing their decision boundaries to be as close to the HIS decision boundaries as possible. Conflict resolution for the HIS is then the determination of the elements for α and β. In the matter of fact, this becomes an ordinary optimization problem, which can be solved by using any optimization method. Due to an advantage of not requiring a derivative calculation and unlikely to be trapped at local minima, GA can be adopted for this purpose. The GA searches for weights α and β that adapt the decisions boundaries of the two modules to be closer together. The hybrid decision finally will be forced to the diagonal path indicated as a dashed line in Figure 11, which ideally shows that the ILFN and the FES are not conflicting and both make correct decisions.

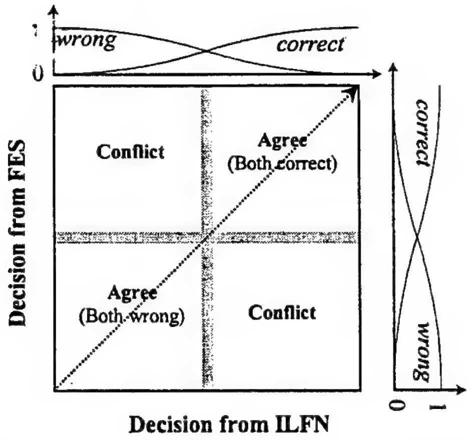


Figure 11: Conflict Decision between the ILFN and the FES

F. Increment Learning Characteristic of the Proposed HIS

An incremental learning system updates its new knowledge without training old data. Only new data is needed in the learning process. This concept has been studied by many researchers (see [35]-[38].) In the hybrid structure between a low level and a higher level, such as in numerical and symbolic or numerical and linguistic systems, it is preferable to incorporate the incremental feature to the system. It is important in application such as controls and monitoring process, as well as in medical diagnosis, to employ an incremental learning aspect. New knowledge needs to be captured in real time without spending tremendous time to learn all the old data along with the new data.

Since the proposed HIS incorporated the ILFN which is an incremental learning architecture, it is easy to employ its incremental learning capability. The ILFN can learn all patterns within only one pass. While operating, the ILFN detects new unseen class prototypes. If new knowledge is found, the new knowledge is added in the hidden unit without destroying the old knowledge. To extend the incremental learning feature to the higher-level linguistic model is straightforward. New linguistic rules can be directly extracted from the new hidden nodes of the ILFN by using the ilfn2rule in the network-to-rule module. An algorithm for checking conflict is operated to maintain consistency between the two levels. Similarly, if the higher level has a new knowledge, i.e., linguistic rules that maybe come from an expert or experienced users, the new knowledge needed to be mapped to the ILFN structures as well. This can be done by using the rule2ilfn algorithm in the rule-to-network module.

III. SIMULATION RESULTS

To demonstrate the performance of the HIS, computer simulations were used in our study. Simulations and analysis of the HIS are performed using the well-known benchmark data, namely Wisconsin breast cancer database (WBCD) [50], as an example for application in medical diagnosis. The WBCD is a real application that has been used by many researchers [12], [23], [26], [29].

The WBCD contains a collection of 699 patterns each described by 9 features. Each feature is a real number in the interval 1 to 10 based on a fine needle aspirate taken directly from human breasts: clump thickness, size uniformity, shape uniformity, marginal adhesion, cell size, bare nuclei, bland chromatin, normal nucleoli and mitosis. The larger the values of these attributes yield the greater the likelihood of malignancy. There are 458 patterns for benign (labeling as "2" in the data base) and 241 patterns for malignant (labeling as "4"). There are 16 patterns with incomplete feature descriptions marked as "?" [50], [51]. We replaced the missing values with "0."

A. Simulation Results for the WBCD

Ten simulations were performed to evaluate the proposed method. In every simulation, the ILFN learning parameters were set to defaults as follows: the threshold, ε, = 0 and the standard deviation, σ_0, = 0.5. The numerical weights of the ILFN network were extracted to fuzzy initial linguistic rules. In order to optimize the linguistic rules, the GA with the integer chromosome representation was used by setting its learning parameters heuristically as follows: population size = 100, the number of generations = 100, the mutation probability, $p_m = 0.8$, and the crossover probability, $p_c = 0.01$. The weights in the fitness evaluation are set as follows: $W_{PC} = 50$, $W_F = 5$, and $W_{NL} = 1$. The number of linguistic labels was constrained to within 3 for each dimension. The GA with a real chromosome representation also was used to find the weighting parameters, α and β. The parameters for the real GA were as follows: population size = 60, the number of generations = 20, the mutation probability, $p_m = 0.8$, and the crossover probability, $p_c = 0.01$. The results from the ten simulations are shown in Table 1.

TABLE 1
The Simulation Results for the WBCD

Run no.	Methods	Structure complexity		Number of patterns		% Correctly classified patterns		
		#Nodes and/or #Rules	# conditions*	# Training	# Test	Training set	% Test set	% Overall patterns
1	Numerical (ILFN)	3 nodes	9 F/N	100	599	98%	96.83%	97.00%
	Fuzzy Rules	3 rules	2.3 F/R	100	599	98%	96.49%	96.71%
	Hybrid ILFN and Fuzzy Rules	3 nodes & 3 rules	9 F/N & 2.3 F/R	100	599	98%	96.83%	97.00%
2	Numerical (ILFN)	3 nodes	9 F/N	100	599	98%	96.83%	97.00%
	Fuzzy Rules	3 rules	4 F/R	100	599	98%	94.74%	96.25%
	Hybrid ILFN and Fuzzy Rules	3 nodes & 3 rules	9 F/N & 4 F/R	100	599	98%	97.14%	97.57%
3	**Numerical (ILFN)**	**3 nodes**	9 F/N	**100**	**342**	**98%**	**97.95%**	**97.23%**
	Fuzzy Rules	3 rules	2.7 F/R	341	342	97.95%	96.49%	96.71%
	Hybrid ILFN and Fuzzy Rules	**3 nodes & 3 rules**	**9 F/N & 2.7 F/R**	**341**	**342**	**98%**	**98.25%**	**98.13%**
4	Numerical (ILFN)	4 nodes	9 F/N	120	358	95.83%	97.49%	96.57%
	Fuzzy Rules	4 rules	3.75 F/R	341	358	97.36%	96.65%	97.00%
	Hybrid ILFN and Fuzzy Rules	4 nodes & 4 rules	9 F/N & 3.75 F/R	341	358	97.07%	97.50%	97.43%
5	Numerical (ILFN)	4 nodes	9 F/N	120	342	95.83%	98.25%	96.93%
	Fuzzy Rules	3 rules	2.67 F/R	341	342	96.77%	96.78%	96.79%
	Hybrid ILFN and Fuzzy Rules	4 nodes & 3 rules	9 F/N, 2.67 F/R	341	342	96.48%	98.25%	97.36%
6	Numerical (ILFN)	5 nodes	9 F/N	150	342	94.67%	97.19%	96.63%
	Fuzzy Rules	**5 rules**	**2.2 F/R**	**341**	**342**	**97.07%**	**97.37%**	**97.22%**
	Hybrid ILFN and Fuzzy Rules	5 nodes, 5 rules	9 F/N & 2.2 F/R	341	342	97.07%	97.08%	97.07%
7	Numerical (ILFN)	5 nodes	9 F/N	150	342	94.67%	97.19%	96.63%
	Fuzzy Rules	4 rules	2.5 F/R	683	683	97.07%	97.07%	97.07%
	Hybrid ILFN and Fuzzy Rules	5 nodes & 4 rules	9 F/N & 2.5 F/R	683	683	97.22%	97.22%	97.22%
8	Numerical (ILFN)	3 nodes	9 F/N	100	342	98%	97.95%	97.23%
	Fuzzy Rules	2 rules	3 F/R	683	683	97.23%	97.23%	97.23%
	Hybrid ILFN and Fuzzy Rules	3 nodes & 2 rules	9 F/N & 3 F/R	683	683	97.57%	97.57%	97.57%
9	Numerical (ILFN)	5 nodes	9 F/N	150	549	94.67%	97.19%	96.42%
	Fuzzy Rules	4 rules	2.5 F/R	699	699	97.57%	97.57%	97.57%
	Hybrid ILFN and Fuzzy Rules	5 nodes & 4 rules	9 F/N & 2.5 F/R	699	699	97.57%	97.57%	97.57%
10	Numerical (ILFN)	3 nodes	9 F/N	100	599	98%	96.83%	97.00%
	Fuzzy Rules	3 rules	2.33 F/R	699	699	96.85%	96.85%	96.85%
	Hybrid ILFN and Fuzzy Rules	3 nodes & 3 rules	9 F/N & 2.33 F/R	699	699	97.42%	97.42%	97.42%
Average	Numerical (ILFN)	3.8 nodes	9 F/N			**96.17%**	**97.37%**	**96.77%**
	Fuzzy Rules	3.4 rules	2.77 F/R			**97.43%**	**96.72%**	**97.08%**
	Hybrid ILFN and Fuzzy Rules	3.8 nodes & 3.4 rules	9 F/N & 2.77 F/R			**97.61%**	**97.48%**	**97.55%**

* F/N = # features per node and F/R = # features per rule.

From Table 1, the ILFN achieved an average correct classification of 96.17% on training set and 97.37% on test set. The fuzzy rules extracted from the trained ILFN achieved an average correct classification of 97.43% on training set and 96.72% on test set. It is worth noting that the fuzzy rules extracted from the trained ILFN achieved higher classification rate for the training set. However, the fuzzy rules achieved lower percentage of correctly classified patterns from the test set. When we combined the ILFN and fuzzy rules extracted to construct a HIS, the results show that the HIS achieved higher classification rate than both the ILFN and the extracted fuzzy rules alone. The proposed HIS had an average of 97.61% and 97.48% correct classification on the training set and the test set, respectively.

Due to the space limitation, we show the details of numerical weights of the ILFN and the extracted linguistic rules from one example based on the best classification performance of the HIS, i.e., from run number 3 in Table 1. Based on run number 3, the details on ILFN and its linguistic rules extracted are shown in Tables 2, 3, and 4.

From run number 3, we used 100 patterns for training the ILFN, 341 patterns for training to the FES and HIS, and used 342 patterns for testing in all three systems. The ILFN constructed 3 hidden nodes with the parameters shown in Table 2. The ILFN network achieved 98% and 97.95% correct classification for the training set and the test set, respectively.

From Table 2, the knowledge embedded in the trained ILFN is in numerical form. Linguistic rules are preferably extracted from the trained ILFN for a reasoning purpose. The fuzzy linguistic rules are mapped from the ILFN parameters and the GA is used to select only discriminatory features. This will be resulted in a more compact rule set.

TABLE 2:
ILFN Parameters for the WBCD

WP									WT
2.7818	1.3455	1.4182	1.2727	2.0545	1.5273	2.7818	1.1818	1.0909	2
7.3462	6.6538	6.6154	5.1923	6.7692	7.5	5.5385	6.9615	3.4615	4
6.6316	3.7895	4.3684	2.6842	3.8947	4	3.8947	4.4211	2.0526	4
Standard Deviation									count
8.0293	3.4577	4.1716	2.6234	2.3358	5.2169	7.9414	2.2247	1.6642	55
8.9208	9.4657	7.9145	12.538	9.2651	9.8717	7.2446	10.184	13.132	26
8.9898	4.0137	4.2122	4.8625	5.2584	7.6044	3.4663	8.9787	7.9811	19

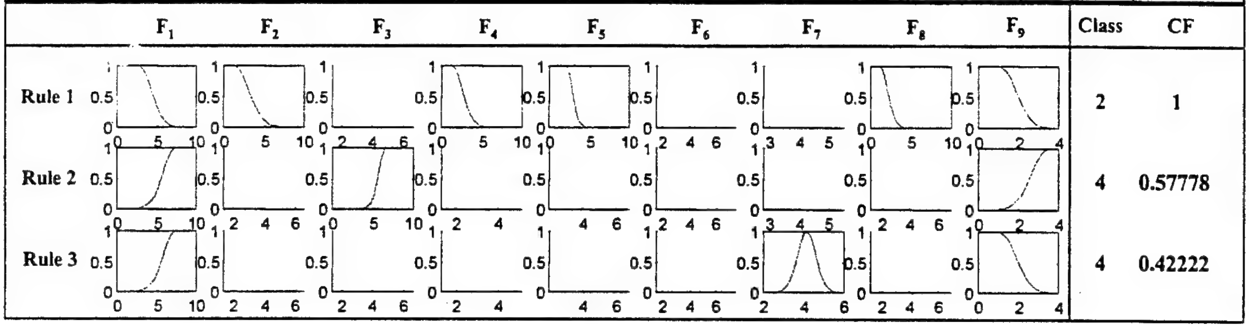
TABLE 3:
Resulted Linguistic Labels and Their Parameters for WBCD

Features	Linguistic Labels and Parameters		
F_1 = Clump Thickness	1: low_1	2: $high_1$	
	(Gaussian: 2.7818, 1.504)*	(Gaussian: 7.3462, 1.504)	
F_2 = Size Uniformity	1: low_2	2: $high_2$	
	(Gaussian: 1.3455, 1.7491)	(Gaussian: 6.6538, 1.7491)	
F_3 = Shape Uniformity	1: low_3	2: $medium_3$	3: $high_3$
	(Gaussian: 1.4182, 0.85625)	(Gaussian: 4.0168, 0.85625)	(Gaussian: 6.6154, 0.85625)
F_4 = Marginal Adhesion	1: low_4	2: $high_4$	
	(Gaussian: 1.2727, 1.2915)	(Gaussian: 5.1923, 1.2915)	
F_5 = Cell Size	1: low_5	2: $medium_5$	3: $high_5$
	(Gaussian: 2.0545, 0.77676)	(Gaussian: 4.4119, 0.77676)	(Gaussian: 6.7692, 0.77676)
F_6 = Bare Nuclei	1: low_6	2: $high_6$	
	(Gaussian: 1.5273, 1.968)	(Gaussian: 7.5, 1.968)	
F_7 = Bland Chromatin	1: low_7	2: $medium_7$	3: $high_7$
	(Gaussian: 2.7818, 0.45416)	(Gaussian: 4.1601, 0.45416)	(Gaussian: 5.5385, 0.45416)
F_8 = Normal Nucleoli	1: low_8	2: $medium_8$	3: $high_8$
	(Gaussian: 1.1818, 0.95222)	(Gaussian: 4.0717, 0.95222)	(Gaussian: 6.9615, 0.95222)
F_9 = Mitosis	1: low_9	2: $high_9$	
	(Gaussian: 1.0909, 0.78113)	(Gaussian: 3.4615, 0.78113)	

* Since Gaussian membership functions are used, the parameters of the lingusistic lables are written as (Gaussian: mean, standard deviation).

TABLE 4:
Fuzzy Expert Rules for Wisconsin Breast Cancer Data

Antecedent									Consequent	
F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	Class	CF
1	1	0	1	1	0	0	1	1	2	1
2	0	3	0	0	0	0	0	2	4	0.57778
2	0	0	0	0	0	2	0	1	4	0.42222



After running for 100 generations, the resulting fuzzy linguistic rules are shown in Table 3. The fuzzy linguistic rules are shown in Table 4. Using the linguistic knowledge from Tables 3 and 4 as the rule set for a fuzzy expert system, the final fuzzy linguistic rules achieved 97.95% and 96.49% correct classification for training and testing data, respectively. The hybrid intelligent system combining the decisions from both ILFN and FES achieved 98% correct classification rate for training set and 98.25% correct classification rate for the testing set. The HIS achieved 98.13% in all 683 patterns of the WBCD. Fuzzy expert rules in natural language for the WBCD can be interpreted as

Rule1: If Clump Thickness is low_1 and Size Uniformity is low_2 and Marginal Adhesion is low_4 and Cell Size is low_5 and Normal Nucleoli is low_8 and Mitosis is low_9, Then Malignant, with confidence = 1;

Rule2: If Clump Thickness is $high_1$ and Shape Uniformity is $high_3$ and Mitosis is $high_9$, Then Benign, with confidence = 0.58;

Rule3: If Clump Thickness is $high_1$ and Bland Chromatin is $medium_7$ and Mitosis is low_9, Then Benign, with confidence = 0.42;

B. Comparison Results Among Other Methods

Several groups of researchers have studied and developed knowledge-based system for the WBCD. Peña-Reyes and Sipper used a fuzzy if-then system as a classifier. They developed a fuzzy-GA algorithm to extract rules from the WBCD. Fuzzy-GA algorithm uses the genetic algorithm (GA) to search for two parameters, P and d, of their fuzzy rules [12]. The number of rules has to be predetermined in *ad hoc* manner. In [23], Setiono developed a rule extraction called NeuroRule. NeuroRule uses a pruning procedure after the training phase to decrease the number of the network connections. The pruning process runs until network performance drops to 95% correct classification rate. In [23], 100-MLP networks were used in the training phase. The network with highest performance out of 100 pruned networks was use in rule-extraction phase. The NeuroRule extracts rules by clustering the hidden nodes activation values. Then, the input combinations are checked if any input makes the hidden nodes and output node active. An improvement of NeuroRule in the WBCD was studied by the same author in [29] by doing data pre-process before the training step. Another group was Taha and Ghosh [26]. In [26], three rule extraction algorithms were developed: BIO-RE, Partial-RE, and Ful-RE. BIO-RE is a black box rule extraction technique which does not require information regarding the internal network structure to generate rules. Partial-RE searches for a set of incoming connections that will cause a unit to be active. Full-RE decompose the rule extraction process into two steps: rules between hidden and output units and rules between input units and hidden units. This is similar to NeuroRule [29] but the difference is that Full-RE employs linear programming and an input discretization method to find a combination of the input values that will cause a hidden unit to be active. Comparison results on the WBCD are shown in Table 5, which shows the comparison among several rule-based systems from [12], [23], [26], [29].

TABLE 5:
Comparison Results for the WBCD Among Well-known Methods

Methods	Representaion Type	Rule complexity		Number of patterns		Performance Evaluation		
		# Rules	# F/R*	# Training	# Test	% Training corr.	% Test corr.	% Overall corr.
Setiono [23]	Boolean Rules	1 + default	2	350	349	96.86%	93.98%	95.42%
Setiono [23]	Boolean Rules	2 + default	4	350	349	97.71%	96.56%	97.14%
Setiono [29]	Boolean Rules	1 + default	4	341	342	97.07%	97.66%	97.36%
Setiono [29]	Boolean Rules	3 + default	3.7	341	342	97.95%	98.25%	98.10%
Setiono [29]	Boolean Rules	4 + default	1	341	342	97.07%	97.66%	97.36%
Setiono [29]	Boolean Rules	5 + default	4.2	341	342	98.53%	97.95%	98.24%
Setiono [29]	Boolean Rules	6 + default	1.7	341	342	97.95%	98.25%	98.10%
Pena and Sipper [12]	Fuzzy Rules	1 + default	4	341	342			97.07%
Pena and Sipper [12]	Fuzzy Rules	2 + default	3	341	342			97.36%
Pena and Sipper [12]	Fuzzy Rules	3 + default	4.7	341	342			97.80%
Pena and Sipper [12]	Fuzzy Rules	4 + default	4.8	341	342			97.80%
Pena and Sipper [12]	Fuzzy Rules	5 + default	3.4	341	342			97.51%
Taha and Ghosh [26]	Boolean Rules	11 + default	2.7	341	342	97.07%	96.20%	96.63%
Taha and Ghosh [26]	Boolean Rules	9 + default	2.67	341	342	97.07%	95.91%	96.49%
Taha and Ghosh [26]	Boolean Rules	5	1.8	341	342	96.77%	95.61%	96.19%
This study	Numerical (ILFN)	(3 nodes)	(9 F/N)	100	342	98%	97.95%	97.23%
	Fuzzy Rules	3	2.7	341	342	97.95%	96.49%	96.71%
	Hybrid ILFN and Fuzzy Rules	(3 nodes & 3 rules)	(9 features & 2.7 F/R)	341	342	98%	98.25%	98.13%

* F/R = # features per rule and F/N = # features per node.

From Table 5, the best performance was from NeuroRule [29] with 5 rules plus a default rule extracted from one of the 100 pruned networks with 2 hidden units and 9 connections. The accuracy rate was 98.24% in 683 patterns. The rule set extracted in [29] is as follows:

If $F_2 \leq 4$ and $F_6 \leq 2$ and $F_8 \leq 2$, then benign,
Else if $F_2 \leq 4$ and $F_6 \leq 2$ and $F_8 \leq 8$ and $F_1 \leq 6$, then benign,
Else if $F_1 \leq 5$ and $F_4 \leq 4$ and $F_6 \leq 5$ and $F_8 \leq 2$, then benign,
Else if $F_1 \leq 6$ and $F_2 \leq 4$ and $F_6 \leq 6$ and $F_8 \leq 8$, then benign,
Else if $F_2 \leq 4$ and $F_4 \leq 5$ and $F_6 \leq 5$ and $3 \leq F_6 \leq 5$ and $F_8 \leq 8$, then benign,
Else malignant

NeuroRule does not produce any rule for malignancy. It needs a default rule for malignancy. Fuzzy-GA [12] extracted rules based on the predetermined number of rules in the range of 1 to 5. A total of 120 evolutionary runs were performed. The highest performance system was 97.80% correct classification rate using 3 fuzzy if-then rules with 4.7 conditions per rule, and a default rule. In [26], using Bio-RE algorithm, the best performance system was 96.96% using 11 Boolean rules with 2.7 conditions per rule. Using Full-RE algorithm, the best performance was 96.19% with 5 rules and 1.8 conditions per rule (no default rule). NeuroRule

[23], [29], fuzzy-GA [12], Bio-RE [26], and Partial-RE [26] have a default rule that means they lack of completeness. Default rules do not provide a symbolic interpretation of the decision other than that "because none of the above occurred" [26].

Based on ten runs, our proposed HIS achieved 98.13% correct classification for all 683 patterns. The HIS used ILFN with 3 hidden nodes and FES with 3 fuzzy if-then rules and 2.7 conditions per rules. An advantage of the proposed HIS is that it incorporates an incremental learning characteristic in the system. Since data can be made available on a daily basis, using the proposed HIS the novel data can be added into the system quickly without spending too much time on retraining all the old information.

IV. CONCLUSION

Combination of a trained ILFN network and a fuzzy expert system (FES) into a unified structure results in a "Hybrid Intelligent System" (HIS) useful for decision-making frameworks. The proposed HIS offers mutually complementary advantages from an ILFN network and a FES. This system can be useful for complex real-world applications, in particular, medical diagnosis where different processing strategies have to be supported.

A mapping mechanism from high-level linguistic knowledge to a low level ILFN network is provided to continuously maintain consistency between low-level and higher-level modules. This allows an expert to add or revise linguistic rules to the system. New knowledge is then mapped back to the ILFN structure allowing the ILFN network to update its parameters.

Fuzzy rules in the FES are generated directly from the ILFN network using the "ilfn2rule" algorithm. After using the ilfn2rule algorithm to map the ILFN numerical variables to linguistic variable, the genetic algorithm is used to improve the rule set. Based on the initial rules extracted, the number of rules and features of the FES are optimized by using the genetic algorithm. A compact FES with only essential discriminatory features is obtained.

The trained ILFN and the optimized FES is combined into a hybrid system. It is found that the combination of the optimized rule base with the trained ILFN achieves better classification results both on training and testing patterns. It is also found that some rules in the original rule set extracted from the trained ILFN may conflict with each other. However, after using the genetic algorithm to refine rules and features, the rule confliction can be resolved.

The resulting knowledge from the proposed rule extraction procedure is represented in "if-then" linguistic form that is easily comprehensible to human users. By integrating the FES and ILFN, explanations and answers can be easily generated when needed while numerical accuracy is maintained.

Computer simulations using the well-known Wisconsin breast cancer database were performed. The low-level ILFN has only few hidden nodes and the higher-level linguistic model extracted has very small number of rules. The trained ILFN and the fuzzy linguistic rules are combined to a HIS achieved very good results based on the classification performance compared with the original system as well as other rule-based methods.

REFERENCES

[1] L. P. Seka, A. Fresnel, D. Delamarre, C. Riou, A. Burgun, B. Pouliquen, R. Duvauferrier, and P. Le Beux, "Computer Assisted Medical Diagnosis Using the Web," *International Journal of Medical Informatics*, Vol. 47, No. 1-2, pp. 51-56, 1997.

[2] F. M. H. M. Dupuits, A. Hasman, and P. Pop, "Computer-Based Assistance in Family Medicine," *Computer Methods and Programs in Biomedicine*, Vol. 55, No. 1, pp. 39-50, 1998.

[3] L. Makris, I. Kamilatos, E. V. Kopsacheilis, and M. G. Strintzis, "Teleworks: A CSCW Application for Remote Medical Diagnosis Support and Teleconsultation," *IEEE Transactions on Information Technology in Biomedicine*, Vol. 2, No. 2, pp. 62-73, 1998.

[4] D. Conforti and L. D. Luca, "Computer Implementation of a Medical Diagnosis Problem by Patterns Classification," *Future Generation Computer Systems*, Vol. 15, No. 2, pp. 287-292, 1999.

[5] D. J. Foran, D. Comaniciu, P. Meer, and L. A. Goodell, "Computer-Assisted Discrimination Among Malignant Lymphomas and Leukemia Using Immunophenotyping, Intelligent Image Repositories, and Telemicroscopy," *IEEE Transactions of Information Technology in Biomedicine*, Vol. 4, No. 4, pp. 265-273, 2000.

[6] K. P. Adlassnig, "The Section on Medical Expert and Knowledge-Based Systems at the Department of Medical Computer Sciences of the University of Vienna Medical School," *Artificial Intelligence in Medicine*, Vol. 21, No. 1-3, pp. 139-146, 2001.

[7] G. P. K. Economou, D. Lymberopoulos, E. Karvatselou, and C. Chassomeris, "A New Concept Toward Computer-Aided Medical Diagnosis—A Prototype Implementation Addressing Pulmonary Diseases," *IEEE Transactions on Information Technology in Biomedicine*, Vol. 5, No. 1, pp. 55-66, 2001.

[8] B. G. Buchana and E. H. Shortcliffe, *Rule-Based Expert Systems: The MYCIN Experiment of the Stanford Heuristic Programming Project*. Reading, Mass.:Addison-Wesley, 1984.

[9] W. A. J. J. Wiegerinck, H. J. Kappen, E. W. M. T. ter Braak, W. J. P. P. ter Burg, M. J. Nijman, Y. L. O, and J. P. Neijt, "Approximate Inference for Medical Diagnosis," *Pattern Recognition Letters*, Vol. 20, No. 11-13, pp. 1231-1239, 1999.

[10] B. Kovalerchuk, E. Vityaev, and J. F. Ruiz, "Consistent Knowledge Discovery in Medical Diagnosis," *IEEE Engineering in Medicine and Biology*, Vol. 19, No. 4, pp. 26-37, 2000.

[11] L. Yan and D. J. Miller, "General Statistical Inference for Discrete and Mixed Spaces by an Approximate Application of the Maximum Entropy Principle," *IEEE Transactions on Neural Networks*, Vol. 11, No. 3, pp. 558-573, 2000.

[12] C. A. Peña-Reyes and M. Sipper, "Designing Breast Cancer Diagnostic via a Hybrid Fuzzy-Genetic Methodology," in *Proceedings of the 1999 IEEE International Fuzzy Systems Conference*, 1999, pp. 135-139.

[13] D. Walter and C. K. Mohan, "ClaDia: a Fuzzy Classifier System for Disease Diagnosis," in *Proceedings of the 2000 Congresss on Evolutionary Computation*, 2000, pp. 1429-1435.

[14] S. Zahan, "A Fuzzy Approach to Computer-Assisted Myocardial Ischemia Diagnosis," *Artificial Intelligence in Medicine*, Vol. 21, No. 1-3, pp. 271-275, 2001.

[15] N. Belacel, Ph. Vincke, J. M. Scheiff, and M. R. Boulassel, "Acute Leukemia Diagnosis Aid Using Multicriteria Fuzzy Assignment Methodology," *Computer Methods and Programs in Biomedicine*, Vol. 64, No. 2, pp. 145-151, 2001.

[16] A. Durg, W. V. Stoecker, J. P. Cookson, S. E. Umbaugh, and R. H. Moss, "Identification of Variegated Coloring in Skin Tumors: Neural Network vs. Rule-Based Induction Methods," *IEEE Engineering in Medicine and Biology Magazine*, Vol. 12, No. 3, pp. 71-74, 98, 1993.

[17] C. S. Pattichis, C. N. Schizas, and L. T. Middleton, "Neural Network Models in EMG Diagnosis," *IEEE Transactions on Biomedical Engineering*, Vol. 42, No. 5, pp. 486-496, 1995.

[18] X. Yao and Y. Liu, "A New Evolutionary System for Evolving Artificial Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 8, No. 3, pp. 694-713, 1997.

[19] D. West and V. West, "Model Selection for a Medical Diagnositic Decision Support System: A Breast Cancer Detection Case," *Artificial Intelligence in Medicine*, Vol. 20, No. 3, pp. 183-204, 2000.

[20] V. Podgorelec, P. Kokol, and J. Zavrsnik, "Medical Diagnosis Prediction Using Genetic Programming," in *Proceedings of the 12th IEEE Symposium on Computer-Based Medical Systems*, 1999, pp. 202-207.

[21] L. W. Man, L. Wai, S. L. Kwong, S. N. Po, and J. C. Y. Cheng, "Discovering Knowledge From Medical Databases Using Evolutionary Algorithms," *IEEE Engineering in Medicine and Biology*, Vol. 19, No. 4, pp. 45-55, 2000.

[22] C. A. Peña-Reyes and M. Sipper, "Evolutionary Computation in Medicine: an Overview," *Artificial Intelligence in Medicine*, Vol. 19, No. 1, pp. 1-23, 2000.

[23] R. Setiono and H. Liu, "Symbolic Representation of Neural Networks," *IEEE Computer*, Vol. 29, No. 3, pp. 71-77, 1996.

[24] A. H. Tan, "Cascade ARTMAP: Integrating Neural Computation and Symbolic Knowledge Processing," *IEEE Transactions on Neural Networks*, Vol. 8, No. 2 , pp. 237-250, 1997.

[25] A. B. Tickle, R. Andrews, M. Golea, and J. Diederich, "The Truth Will Come to Light: Directions and Challenges in Extracting the Knowledge Embedded within Trained Artificial Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 9, pp. 1057-1068, 1998.

[26] I. A. Taha and J. Ghosh, "Symbolic Interpretation of Artificial Neural Networks," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, No. 3, pp. 448-463,1999.

[27] P. Tino and M. Koteles, "Extracting Finite-State Representations from Recurrent Neural Networks Trained on Chaotic Symbolic Sequences," *IEEE Transactions on Neural Networks*, Vol. 10, No. 2, pp. 284-302, 1999.

[28] S. Mitra and Y. Hayashi, "Neuro-Fuzzy Rule Generation: Survey in Soft Computing Framework," *IEEE Transactions on Neural Networks*, Vol. 11, No. 3, pp. 748-768, 2000.

[29] R. Setiono, "Generating Concise and Accurate Classification Rules for Breast Cancer Diagnosis," *Artificial Intelligence in Medicine*, Vol. 18, No. 3, pp.205-219, 2000.

[30] Y. Hayashi, "A Comparison Between Two Neural Network Rule Extraction Techniques for the Diagnosis of Hepatobiliary Disorders," *Artificial Intelligence in Medicine*, Vol. 20, No. 3, pp. 205-216, 2000.

[31] S. I. Gallant, *Neural Network Learning and Expert Systems*. Cambridge, MA: MIT Press, 1993.

[32] L. R. Medsker, *Hybrid Neural Network and Expert Systems*. Boston: Kluwer Academic Publishers, 1994.

[33] S. Goonatilake and S. Khebbal, *Intelligent Hybrid Systems*. Chichester: Wiley, 1995.

[34] S. Wermter and R. Sun, "An Overview of Hybrid Neural System," in *Hybrid Neural Systems*, S. Wermter and R. Sun, Eds., Berlin: Springer-Verlag, 2000, pp.1-13.

[35] L. M. Fu, H. H. Hsu, and J. C. Principe, "Incremental Backpropagation Learning Networks," *IEEE Transactions on Neural Networks*, Vol. 7, No. 3, pp. 757-761, 1996.

[36] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps," *IEEE Transactions on Neural Networks*, Vol. 3, No. 5, pp. 698-713, 1992.

[37] G. Yen and P. Meesad, "Pattern Classification by an Incremental Learning Fuzzy Neural Network," in *Proceedings of the International Joint Conference on Neural Networks,* 1999, pp. 3230-3235.

[38] G. Yen and P. Meesad, "An Effective Neuro-Fuzzy Paradigm for Machinery Condition Health Monitoring," to appear *in IEEE Transactions on System, Man, and Cybernetics, Part B: Cybernetics*, 2001.

[39] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and Its Application to Modeling and Control," *IEEE Transactions on System, Man, Cybernetics*, Vol. 15, No. 1, pp. 116-132, 1985.

[40] J. H. Holland, *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI: University of Michigan Press, 1975.

[41] L. A. Zadeh, “Fuzzy Sets,” *Information and Control*, Vol. 8, pp. 338-353, 1965.

[42] Y. Jin, “Fuzzy Modeling of High-Dimensional Systems: Complexity Reduction and Interpretability Improvement,” *IEEE Transactions on Fuzzy Systems*, Vol. 8, No. 2, pp. 212-221, 2000.

[43] M. Setnes, R. Babuska, and H. B. Verbruggen, “Rule-Based Modeling: Precision and Transparency,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 28, No. 1, pp. 165-169, 1998.

[44] H. Ishibuchi, T. Nakashima, and T. Murata, “Performance Evaluation of Fuzzy Classifier Systems for Multidimensional Pattern Classification Problems,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 29, No. 5, pp. 601-618, 1999.

[45] Y. Jin, W. V. Seelen, and B. Sendhoff, “On Generating FC^3 Fuzzy Rule Systems from Data Using Evolution Strategies,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 29, No. 6, pp. 829-845, 1999.

[46] O. Cordon and F. Herrera, “A Proposal for Improving the Accuracy of Linguistic Modeling,” *IEEE Transactions on Fuzzy Systems*, Vol. 8, No. 3, pp.335-344, 2000.

[47] L. X. Wang and J. M. Mendel, “Generating Fuzzy Rules by Learning from Examples,” *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 22, No. 6, pp. 1414-1427, 1992.

[48] M. Setnes, R. Babuska, U. Kaymak, and H. R. van Nauta Lemke, “Similarity Measures in Fuzzy Rule Base Simplification,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 28, No. 3, pp. 376-386, 1998.

[49] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[50] W. H. Wolberg and O. L. Mangasarian, “Multi-surface Method of Pattern Separation for Medical Diagnosis Applied to Breast Cytology,” *Proceedings of the National Academy of Sciences, U.S.A.*, Vol. 87, pp. 9193-9196, 1990.

[51] C. L. Blake and C. J. Merz, *UCI Repository of Machine Learning Databases* [http://www.ics.uci.edu/~mlearn/MLRepository.html], Irvine, CA: University of California, 1998.

APPENDIX F:

Reconfigurable Control System Design For Fault Diagnosis and Accommodation

by

Liang-Wei Ho and Gary G. Yen

Submitted to
International Journal of Control

RECONFIGURABLE CONTROL SYSTEM DESIGN FOR FAULT DIAGNOSIS AND ACCOMMODATION[*]

Liang-Wei Ho *Gary G. Yen*

Intelligent Systems and Control Laboratory
School of Electrical and Computer Engineering
Oklahoma State University

ABSTRACT

The growing demand in system reliability and survivability under failures has urged ever-increasing research effort on the development of fault diagnosis and accommodation. In this research work, the on-line fault tolerant control problem for dynamic systems under unanticipated failures is investigated from a realistic point of view without any specific assumption on type of system dynamical structure or failure scenarios. The necessary and sufficient conditions for system on-line stability under catastrophic failures have been derived using the discrete-time Lyapunov stability theory. Based upon the existing control theory and the modern intelligent techniques, an on-line fault accommodation control strategy is proposed to deal with the desired trajectory-tracking problems for systems suffering from various *unknown* and *unanticipated* catastrophic component failures. Theoretical analysis indicates that the control problem of interest can be solved on-line without a complete realization of the unknown failure dynamics provided an on-line estimator that satisfied certain conditions. Through the on-line estimator, effective control signals to accommodate the dynamic failures can be computed using only the partially available information of the faults. Several on-line simulation studies have been presented to demonstrate the effectiveness of the proposed strategy. To investigate the feasibility of using the developed technique for unanticipated fault accommodation in real hardware under the real-time environment, an on-line fault tolerant control test bed has been constructed to validate the proposed technology. Both on-line simulations and the real-time experiment show encouraging results and promising futures of on-line real-time fault tolerant control based solely upon insufficient information of the system dynamics and the failure modes.

1. INTRODUCTION

The quest for the system reliability under failures has drawn significant attention in the intelligent control community. These failures will usually result into the deviations of the system dynamics, which can be characterized by drastic changes of the system parameters or, more seriously, the inherent dynamical *structure* of the system. The system stability becomes a critical concern. In many safety-critical systems such as aircrafts or nuclear power plants, system stability under failure incidents seriously impacts human survivability. Urging by these growing demands in system safety and reliability, extensive research activities have been focusing on

[*] This work was supported in part by the U.S. Air Force Office of Science Research under Grant F49620-98-1-0049 and National Science Foundation, Measurement and Control Engineering Center.

developing Fault Diagnosis and Accommodation (FDA) or so-called Fault Tolerant Control (FTC) methodology to maintain the system stability and to avoid the loss of human life under various failure scenarios.

The primary objective of FDA is to detect system failures, identify the faults, and eventually accommodate the tolerable faults by *on-line* adjusting the control actions. From the realistic point of view, the ultimate goal of FDA is to at least maintain the system on-line stability under failures. Most of the existing fault accommodation schemes are mainly designed either based upon the powerful and well-developed linear design methodology or based upon the assumption of a certain type of nonlinear systems under simple failure conditions to obtain the desired objectives. The representative approaches include the pseudo-inverse method or model-following method [1], eigenstructure assignment [2], LQC [3], additive compensation for sensor and actuator failures [4], reconfiguration control with parameter identification [5], state-space pole placement [6], and the learning approach [7-8]. However, this is rarely the cases in practice since all the systems are inherently nonlinear and the system dynamics under failure situations are much more likely to be nonlinear and time varying. Although the control law reconfigurations are relatively easy to realize and solve under the assumption of linear structures, the usefulness in practical situations is also limited. Nevertheless, for a dynamic system under totally unanticipated catastrophic failures, it is not a reasonable approach to assume certain types of dynamic changes caused by the unexpected failures. From the realistic on-line fault tolerant control point of view, the interesting question becomes, whether the system under catastrophic failures is still controllable, how to effectively and efficiently detect the failures, identify the change of the system dynamics, and reconfigure or adjust the control actions to accommodate the system failures, maintain the system on-line stability, and possibly recover the system performance by using only the insufficient information in the on-line situation without human intervention.

In this research work, the on-line fault accommodation control problems for nonlinear dynamic systems under catastrophic failures are investigated. The problems are analyzed and approached without any specific assumption on the type of system dynamical structure or certain kind of failure scenarios. The major interest and contribution of this paper is to develop an effective on-line failure accommodation technique for general nonlinear dynamic systems with *unanticipated component* failures in real-world applications. Based upon the discrete-time Lyapunov stability theory, the necessary and sufficient conditions to guarantee system on-line stability and performance under catastrophic failures have been established. A general intelligent on-line fault tolerant control strategy is presented to deal with the on-line fault detection and control law reconfiguration problems. This paper is organized as follows. In Section 2, we define the on-line fault accommodation control problem of interest. In Section 3, the problem is analyzed under the general formulation and the necessary and sufficient conditions to maintain the systems on-line stability are derived through theoretical analysis. An intelligent on-line fault accommodation control strategy together with some implementation issues are proposed in Section 4. The simulation study is provided in Section 5 to show the effectiveness of the proposed on-line control framework in various failure cases. Section 6 is dedicated to demonstrate how the developed strategy can be implemented in commercial-off-the-shelf hardware under the real-time environment. The conclusion is included in Section 7 along with the discussion of future research directions.

2. FTC PROBLEM OF INTEREST

A general n-input, m-output dynamic system can be described by Equation (1)

$$\begin{aligned} & y_l(k+d) = f_l(\bar{y}_1, \bar{y}_2, \ldots \bar{y}_m, \bar{u}_1, \bar{u}_2, \ldots \bar{u}_n), \\ & \bar{y}_i = \{y_i(k+d-1), y_i(k+d-2), \ldots, y_i(k+d-p_i)\}, \\ & \bar{u}_j = \{u_j(k), u_j(k-1), \ldots u_j(k-q_j)\}, \\ & p_i, q_j \in \mathfrak{R}^+, i = 1, 2, \ldots, m., j = 1, 2, \ldots, n., \mathit{and}\ l = 1, 2, \ldots, m., \end{aligned} \tag{1}$$

where $f_l : \mathfrak{R}^P \times \mathfrak{R}^Q \mapsto \mathfrak{R}$, with $P = \sum_{i=1}^{m} p_i$, $Q = \sum_{j=1}^{n} q_j$ is the mathematical realization of the system dynamics for the lth output. $y_l, y_i, u_j \in \mathfrak{R}$ are the lth and ith system outputs and jth input, respectively. d is the relative degree of the system (the smallest delay from the input signal to the system output). In general, f_l may not be readily available in mathematical format all the time due to the difficulty of modeling a complex dynamic system. However, it is possible to develop a realization to describe the system behavior off-line with a known bounded uncertainty within the desired working region of the system using all the existing modeling techniques. These techniques include the state-of-the-art computational intelligence technology such as artificial neural networks or fuzzy logics provided enough computing resource and sufficient time for the development of the realization [9-11] as shown in Equation (2)

$$y_l(k+d) = \hat{f}_l(\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_m, \bar{u}_1, \bar{u}_2, \ldots \bar{u}_n) + \eta_l(y,u), \tag{2}$$

where $\|\eta_l(y,u)\|_2 \le \delta_0$, $\forall (y,u) \subseteq (\mathbf{Y}, \mathbf{U})$, $(\mathbf{Y}, \mathbf{U})$ represents the desired working regime, and $\delta_0 \in \mathfrak{R}$ is a known constant. Thus, $\hat{f}_l$, the realization of the real system with a known bounded uncertainty within the desired working region of the system, will be either a mathematical, numerical, or a combined realization and it is assumed that this realization is developed off-line and available. Equation (1) denotes a healthy system under the fault-free situation and Equation (2) is the corresponding nominal model. Under different component failures, the system dynamics is represented by the following equation

$$\begin{aligned} y_l(k+d) = {} & f_l(\bar{y}_1, \bar{y}_2, \ldots \bar{y}_m, \bar{u}_1, \bar{u}_2, \ldots \bar{u}_n) \\ & + \sum_{v=1}^{r} \beta_v^l(k - T_v^l) F_v^l(\bar{y}_1, \bar{y}_2, \ldots \bar{y}_m, \bar{u}_1, \bar{u}_2, \ldots \bar{u}_n, k), \end{aligned} \tag{3}$$

where $F_v^l(\cdot) : \mathfrak{R}^P \times \mathfrak{R}^Q \times \mathfrak{R}^+ \mapsto \mathfrak{R}$ represents the dynamic change (a general *time-varying* function depends upon past system outputs, past control inputs, and the current control input) caused by the unknown and possibly unanticipated failure mode v for the lth output. $F_v^l(\cdot)$, $\beta_v^l(\cdot)$, and T_v^l are assumed unknown due to the possible occurrence of unanticipated failures. r is the number of system failures. All the cases in which $r > 1$ are referred to as multiple-failure cases. Two typical faults, incipient faults and abrupt faults, are considered to be involved on-line. Their characteristics can be described by the time-varying gains: the incipient fault:

$\beta_v^l(k-T_v^l)=(1-e^{-\alpha_v^l(k-T_v^l)})U(k-T_v^l)$; the abrupt fault: $\beta_v^l(k-T_v^l)=U(k-T_v^l)$. $\alpha_v^l \in \Re^+$ is an unknown constant which defines the time profile of the incipient failure mode v and $U(k)$ denotes the unit step function. Abrupt failures are used to represent the sudden change of the system dynamics due to catastrophic malfunction or failure of the system component while the incipient failures are used to describe the time-varying effect of the system component-aging behavior.

The control objective is to generate appropriate control signals to stabilize the system and, possibly, drive the system outputs back to the desired trajectories, $y_{ld}(k+d) \in \Re, l=1,2,\ldots,m.$, in on-line situations with the presence of the abrupt/incipient faults and modelling uncertainties. Of course, many system failure situations are catastrophic and uncontrollable. For example, if the sensor loop malfunctions such that all the readings from the sensor are lost or meaningless, without knowing the true failure, the only way to possibly maintain the system safety is by having human interference such as shut-down of the system, failure diagnosis, and replacement of the faulty parts. If the failures actually break down the control input to the system, there is no way to perform any control recovery on-line. In on-line situations, the interesting and important question becomes how to properly control the system behavior in time to prevent the failure from causing more serious lost, if the system under failure is still controllable at that time. Figure 1 shows the interest problem region of this research, the curve-line area, where the system behavior under failures are out of the nominal working area, but still controllable and within the physically available working region. The major FTC objective is to prevent the faulty system from moving into the saturation regime and possibly drive it back to the nominal condition.

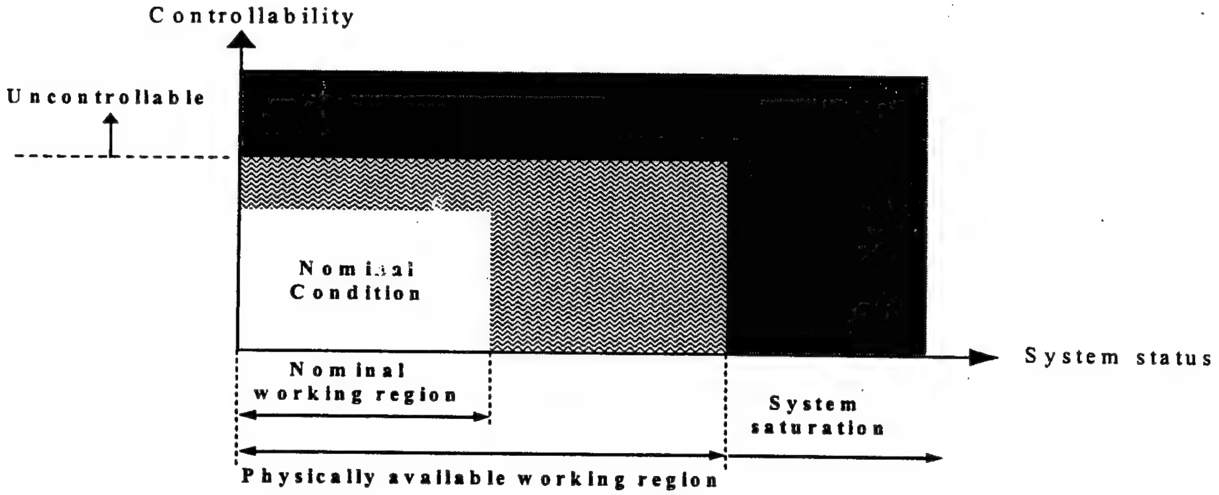


Figure 1. FTC problem region of interest

3. ON-LINE RECONFIGURABLE CONTROL

3.1 Theoretical foundation and analysis

Without loss of the generality, we let $d=1$ and consider the SISO system to facilitate the analysis and derivation. Define a S function representing the desired dynamics as shown in Equation (4)

$$S(k)=\frac{y_d(k)-y_d(k-1)}{\Delta t}-\frac{y(k)-y(k-1)}{\Delta t}+a(y_d(k)-y(k)), \tag{4}$$

where $y_d(k)$ and $y(k)$ represent the desired system output and the actual system output at time step k, respectively. Δt represents the sampling period. a is a positive real number which defines how fast the system output will converge to the desired output. The desired dynamics can then be described by setting the S function equal to zero (i.e., $S=0$) and it is a function of tracking error, $e(k)=y_d(k)-y(k)$. According to the discrete-time Lyapunov stability theory, if we let $V(e(k))=S^2(e(k))$ as the Lyapunov function candidate, the controller design objective becomes satisfying $V(e(k+1))\le V(e(k))$. (To simplify the notation, e will be eliminated from the remaining of this paper.) This implies $S^2(k+1)<S^2(k) \Leftrightarrow [S(k+1)+S(k)][S(k+1)-S(k)]<0$, which is the same as satisfying the following inequalities,

$$\begin{aligned}&-S(k)<S(k+1)<S(k) \quad \textit{when } S(k)>0\\ &S(k)<S(k+1)<-S(k) \quad \textit{when } S(k)<0.\end{aligned} \tag{5}$$

For $S(k)>0$,
plugging in Equation (4), we have

$$-S(k)<\frac{y_d(k+1)-y_d(k)}{\Delta t}-\frac{y(k+1)-y(k)}{\Delta t}+a(y_d(k+1)-y(k+1))<S(k). \tag{6}$$

Reorganizing the inequality, we get

$$-S(k)-\overline{Y}(k)<(-a-\frac{1}{\Delta t})y(k+1)<S(k)-\overline{Y}(k), \tag{7}$$

where $\overline{Y}(k)=\dfrac{y_d(k+1)-y_d(k)+y(k)}{\Delta t}+ay_d(k+1)$. This can be further simplified as

$$(\overline{Y}(k)+S(k))(a+\frac{1}{\Delta t})^{-1}>y(k+1)>(\overline{Y}(k)-S(k))(a+\frac{1}{\Delta t})^{-1}. \tag{8}$$

For $S(k)<0$, we have

$$(\overline{Y}(k)-S(k))(a+\frac{1}{\Delta t})^{-1}>y(k+1)>(\overline{Y}(k)+S(k))(a+\frac{1}{\Delta t})^{-1}. \tag{9}$$

Notice that the left hand side and the right hand side of inequalities (8) and (9) can be computed at each time step. So, the on-line control problem becomes *finding the effective control signal which satisfies inequality (8), when* $S(k)>0$ *or (9), when* $S(k)<0$ *at every time step.*

The true system output, $y(k+1)$, can be approximated by the sum of the outputs from the nominal model and an on-line estimator which is used to realize the unknown failure dynamics as follows:

$$\begin{aligned}y(k+1)&=Ny(k+1)+fy(k+1)\\ &=N\hat{y}(k+1)+N\tilde{y}(k+1)+nfy(k+1)+nf\tilde{y}(k+1),\end{aligned} \tag{10}$$

where $Ny(k+1)$, $fy(k+1)$, $N\hat{y}(k+1)$, $N\tilde{y}(k+1)$, $nfy(k+1)$, and $nf\tilde{y}(k+1)$ denote the nominal system, the failure dynamics, the nominal model, the remaining uncertainty between the nominal system and the nominal model, the on-line estimator, and the remaining uncertainty between the estimator and the failure dynamics, respectively, (i.e., $Ny(k+1) = N\hat{y}(k+1) + N\tilde{y}(k+1)$ and $fy(k+1) = nfy(k+1) + nf\tilde{y}(k+1)$). The effective control input satisfying the inequalities (8) or (9) can be found using optimization algorithms. Modern population-based optimization techniques such as the genetic algorithm, evolutionary strategy, immune algorithm, simulated annealing, etc., have been exploited in a variety of areas and applications [12-15]. However, although the effectiveness in achieving successful optimization objectives has been demonstrated, most of them are applicable in off-line situations at present, due to the time-consuming iterative process. From the computational complexity point of view, the well-known and efficient gradient descent algorithm will be considered and used in the remaining of this paper because of its popularity in on-line applications. The desired point at every time step is

$$\begin{aligned} Desire(k) &= [(S(k) + \bar{Y}(k))(a + \frac{1}{\Delta t})^{-1} + (\bar{Y}(k) - S(k))(a + \frac{1}{\Delta t})^{-1}]/2 \\ &= \bar{Y}(k)(a + \frac{1}{\Delta t})^{-1}. \end{aligned} \tag{11}$$

Define the error as

$$Error(k) = Desire(k) - N\hat{y}(k+1) - N\tilde{y}(k+1) - nfy(k+1) - nf\tilde{y}(k+1). \tag{12}$$

The effective control input can be searched based upon the gradient descent algorithm for square error,

$$\begin{aligned} \frac{\partial Error(k)^2}{\partial u(k)} &= 2Error(k)\frac{\partial Error(k)}{\partial u(k)} \\ &= -2Error(k)[\frac{\partial N\hat{y}(k+1)}{\partial u(k)} + \frac{\partial N\tilde{y}(k+1)}{\partial u(k)} + \frac{\partial nfy(k+1)}{\partial u(k)} + \frac{\partial nf\tilde{y}(k+1)}{\partial u(k)}]. \end{aligned} \tag{13}$$

The resulting control input will be updated by

$$u(k)_{new} = u(k)_{old} - \alpha(k)\frac{\partial Error(k)^2}{\partial u(k)}, \tag{14}$$

where $\alpha(k)$ is a time-varying learning rate parameter. The searching procedure is repeated until the inequalities (8) or (9) holds, the control input converges, or the maximum number of iterations is reached. Of course, the term, $nf\tilde{y}(k+1)$, the remaining uncertainty of the failure dynamics, and $N\tilde{y}(k+1)$, the remaining uncertainty of the nominal system, are unknown and the terms, $\frac{\partial nf\tilde{y}(k+1)}{\partial u(k)}$ and $\frac{\partial N\tilde{y}(k+1)}{\partial u(k)}$, cannot be computed either. So, the actual searching procedure is based upon the approximated values:

$$Er\hat{r}or(k) = Desire(k) - N\hat{y}(k+1) - nfy(k+1), \text{ and} \tag{15}$$

$$\frac{\partial Er\hat{r}or(k)^2}{\partial u(k)} = -2Er\hat{r}or(k)[\frac{\partial N\hat{y}(k+1)}{\partial u(k)} + \frac{\partial nfy(k+1)}{\partial u(k)}]. \tag{16}$$

It can be shown that the S function is bounded as shown in Equation (17) [please refer to Appendix]. Thus, the necessary and sufficient conditions to guarantee the system on-line stability under unanticipated failures is the finiteness of the three terms,

$\sup_{\forall k>T_f}\{\|N\bar{y}(k+1)\|\}$, $\sup_{\forall k>T_f}\{\|nf\bar{y}(k+1)\|\}$, $\sup_{\forall k>T_f}\{\|\Delta Error(k)\|\}$, (i.e., the modeling uncertainty, on-line estimation error, and the optimization error are finite after the time step, T_f, at which time the control actions are reconfigured for proper failure accommodation).

$$\Sigma \le S(k+1) \le \Xi, \tag{17}$$

where

$$\Sigma = -\left[\sup_{\forall k>T_f}\{\|N\bar{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|nf\bar{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|\Delta Error(k)\|\}\right](a+\frac{1}{\Delta t}), \text{ and}$$

$$\Xi = \left[\sup_{\forall k>T_f}\{\|N\bar{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|nf\bar{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|\Delta Error(k)\|\}\right](a+\frac{1}{\Delta t}).$$

The discrete-time Lyapunov stability theory indicates that the control problem can be solved as long as *the numerical value of the failure dynamics* is realized at each time step, which is a measurement of how far the failure drives the system dynamics away from the desired dynamics. Based upon the above theoretical analysis, the system under unexpected catastrophic failures can be stabilized on-line and the performance can be recovered provided an effective on-line estimator for the unknown failure dynamics such that the necessary and sufficient condition is satisfied. Furthermore, since the on-line estimator is used to provide the approximated numerical value of the failure dynamics at each time step based upon the most recent measurements (i.e., the failure may be time-varying), no specific structure or dynamics is required for the estimator. In other words, only a static function approximator that approximates the most recent behavior of the failure is needed for the control purpose. At every time step, the desired point, $\bar{Y}(k)(a+\frac{1}{\Delta t})^{-1}$, is computed, and the effective control signal is searched to ensure that the actual result is as close to the desired point as possible through the realization of the nominal system dynamics and the failure dynamics.

3.2 On-line fault tolerant control laws for a class of nonlinear systems

If the nominal system behavior under the fault-free condition can be described by the so-called companion form or controllability canonical form (i.e., *linear in control*) and the fault is a function of system outputs, the *first* on-line control law can be derived for real-time failure accommodation as shown in Equation (18)

$$u(k) = [\bar{Y}(k)(a+\frac{1}{\Delta t})^{-1} - f(\cdot) - NF(k)]\frac{1}{g(\cdot)}, \tag{18}$$

where the nominal model is in the form of $y(k+1) = f(\cdot) + g(\cdot)u(k)$, and $NF(k)$ denotes the numerical value of the failure dynamics at time step k realized through the on-line estimator. Sharing the similar spirit of the Discrete-time Sliding Mode Control technique [16], the *alternative corrective* control law that possesses more robustness property for the control problems of our interest can be developed as follows:

$$u(k) = u_1(k) + u_2(k), \tag{19a}$$

$$u_2(k) = \frac{K(k)}{D(k)g(\cdot)} sat\left(\frac{S(k)}{\varphi(k)}\right) + U(k - T_c)\frac{-NF(\cdot)}{g(\cdot)}, \tag{19b}$$

where $u_1(k)$ represents the nominal control law and T_c denotes the specific time step at which time the difference of the sum square approximation errors of the on-line estimator during two consecutive windows, Ω, is below a pre-specified threshold, δ, at the first time. This implies that the on-line learning result cannot be further improved significantly (i.e., $\Omega \le \delta$). $\varphi(k)$ denotes the boundary layer thickness. The saturation function is defined to be

$$\mathrm{sat}\left(\frac{S(k)}{\varphi(k)}\right) = \begin{cases} +1 \ , & if\ S(k) > \varphi(k) \\ \dfrac{S(k)}{\varphi(k)}, & if\ |S(k)| \le \varphi(k) \ . \\ -1 \ , & if\ S(k) < -\varphi(k) \end{cases}$$

The boundary thickness and the controller gain are defined as

$$\varphi(k) = \eta(k) + \varepsilon, \tag{20}$$

$$K(k) = \eta(k) + 2\varepsilon, \tag{21}$$

respectively, where ε is an arbitrary positive constant and $\eta(k)$ will be updated using the following equation

$$\eta_{new}(k) = \begin{cases} \sup\limits_{L}\left\{\left|D(k)\Delta\hat{f}(.)\right|\right\} = \sup\limits_{L}\left\{\left|D(k)\left(\sum\limits_{i=1}^{r}\beta_i(\cdot)F_i(\cdot) - NF(\cdot)\right)\right|\right\}, & if\ \Omega \le \delta \\ \eta_{old}\ , otherwise \end{cases}, \tag{22}$$

and

$$D(k) \approx \frac{S(k) - S(k-1)}{\tilde{y}(k) - \tilde{y}(k-1)} \ ; \ \tilde{y}(k) = y_d(k) - y(k). \tag{23}$$

The boundary layer thickness is now redefined by the least upper bound of the failure uncertainty, on-line approximation error, as shown in Equations (20) and (22) by tracking $\sum_{i=1}^{r}\beta_i(\cdot)F_i(\cdot)$ (i.e., multiple failures) on-line. Once a fault is detected, the control signal is adjusted by adding the corrective control signal, $u_2(k)$, to confine the system performance within a boundary layer using the first term of the right hand side in Equation (19). At the same time, an estimator is initialized and learns (approximates) the unknown failure mode dynamics on-line. The on-line learning result is monitored and evaluated by using the following criterion:

$$SSAE0 = \sum_{k=k_0}^{k_0+l-1}(fy(k) - nfy(k))^2 \tag{24a}$$

$$SSAE1 = \sum_{k=k_0+l}^{k_0+2l-1} (fy(k) - nfy(k))^2 \tag{24b}$$

$$\Omega = |SSAE1 - SSAE0| \tag{24c}$$

where $SSAE0$ and $SSAE1$ stand for the sum square approximation errors of the on-line estimator during two consecutive windows. $nfy(k)$ and $fy(k)$ are the output of the estimator and the difference between the measurement and the output of the nominal model at time step k, respectively. The design parameter l represents a time period such that the least upper bound of the approximation error is evaluated every time period, $L = [k - l, k]$. Equations (20)-(22) state that both the boundary layer thickness and the controller gain are automatically estimated and adjusted on-line by the estimator to further reduce control error. Another important point that should be mentioned here is that by adding $U(k - T_c)$ in the second term of the right hand side in Equation (19b) to delay the compensation of the nominal control law until the convergence of the learning result, it is assumed that the system under nominal control law will not lose the stability before time step T_c. This delay usually results in less sensitivity in the selection of the learning rate in the learning process and a better transient performance. The detail of analysis, discussion, and simulation tests of both on-line control laws has been reported in [17-18].

4. ON-LINE FAULT TOLERANT CONTROL STRETAGY

Figure 2 shows the basic framework of the on-line control strategy for the system that may be subject to the unanticipated catastrophic failures. The intelligent control regulator monitors and evaluates system behavior at every time instant through a fault detection mechanism. During the normal operation mode that corresponds to the non-failure situation, the nominal control signal will be passed to the system to control its behavior. Once an abnormal system behavior is detected by the fault detection mechanism, an on-line estimator is initialized and starts learning the unknown failure dynamics. When the learning process converges, the control law is reconfigured and computed by the regulator based upon the current knowledge of the failure dynamics provided by the on-line estimator. The intelligent control regulator also has to interact with the supervisor to accept higher priority commands, such as change of the control objective or design parameters, and warn the supervisor for emergent shut-down of the system in cases that the unanticipated system failures are catastrophic and the system is actually uncontrollable after the failures. Successful fault tolerance mission highly relies upon the off-line design efforts that include the accuracies of the nominal model, controller, and the design parameters of on-line approximator and fault detection scheme. The details are discussed as follows:

4.1 Design of the nominal model

The main design objective of the nominal model is to satisfy Equation (2) within the desired working regime. Specifically, it is at best that this region covers the entire physically available working environment as shown in Figure 1 (i.e., the nominal working region is not necessarily the same as the available working region). Apparently, the more accurate the nominal model is within the available region, the less effort we need to exert to estimate on-line when the system is under failures. It is the possibility that the nominal model can be developed and tested *off-line* makes this goal practically feasible. For complex dynamic systems whose mathematical description of the nominal model is not readily available, the so-called intelligent modelling techniques have shown effectiveness in the complicated modeling process provided sufficient resource and time for the satisfactory development. These techniques include polynomials, rational functions, spline functions, multiplayer perceptron networks, radial-basis-function networks, adaptive fuzzy systems, and HME (Hierarchical Mixture of Expert Networks) [10-11].

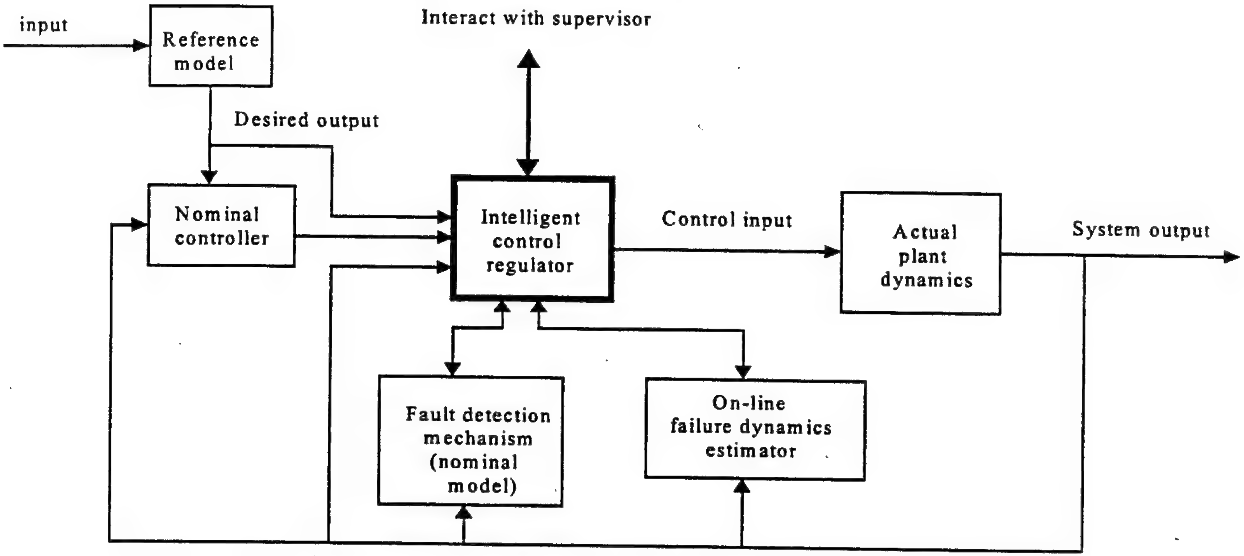


Figure 2 Basic framework of on-line fault tolerant control

4.2 Design of the nominal controller

The second step is to design the nominal controller based upon the nominal model. For the model whose mathematical representation is not in controllability canonical form or the model is realized through the intelligent modelling techniques, the closed-form control law may not be easily formulated. In these cases, neural controllers have been introduced for the control purpose [19-20]. However, despite the fact that the effectiveness of the neural controller has been shown, the design or training process of the controller is still quite complicated since the training of the neural controller involves a dynamic system. Representative approaches include the so-called dynamic backpropagation [19], backpropagation through time [29-31], finite difference approximation, and SPSA (Simultaneous Perturbation Stochastic Approximation) [21]. A more computationally efficient design method of the neural controller can be constructed using the similar approach discussed in Section 3. The details of the design procedure used in the on-line simulation section is described as follows:

a. Set up initial conditions (i.e., the S function in Equation (4), the initial system states, the initial control input, etc.).
b. Compute the desired point at each time step (i.e., Equation (11)) and define the error (i.e., Equation (12) without the failure terms).
c. Compute the Jacobian of the nominal model with respect to the control inputs.
d. Search the effective control inputs to satisfy inequality (8) or (9) using Equations (13)-(16) without the failure terms.

e. Collect the training patterns for the neural controller (i.e., the system outputs and the effective control inputs that satisfy the inequalities).
f. Go back to step b. until the time ends.
g. Train the neural controller with the selected network structure, the collected training patterns, and static training algorithm [22].

This approach breaks the complex training process of the neural controller into the gradient descent optimization and the static training method that appears to be effective and requires much less computational cost.

4.3 On-line fault detection scheme

The next step is to test the performance of the nominal controller. Equation (25) shows a testing criterion that evaluates the mean square control error within a fixed length of window with ω as a design parameter.

$$\psi = \frac{1}{\omega}\sum_{k=k_0}^{k_0+\omega-1}(y_d(k)-y(k))^2 , \tag{25}$$

where k_0 is the starting time step. The testing result that indicates the system behavior under nominal controller in the fault-free situation not only can be used for the evaluation of the design of the nominal controller, but also provide useful information for the on-line fault detection. It is known that the complete on-line fault detection and diagnosis is still impossible at present due to the inherent complexity of the problems and time constraint in on-line situations [6]. An ideal fault detection scheme should be sensitive to detect any fault that deteriorates system performance and robust enough to reject false alarms that are possibly caused by modeling error, disturbance, and measurement noises. Unfortunately, these two goals are usually conflict with each other and the selection of proper fault detection algorithm becomes a trade-off decision. Since the system safety under failures is the first priority of control missions, false alarm is always more preferable than the miss detection. In other words, a conservative fault detection scheme which guarantees the miss-detection-free is more likely to be adopted. From both system safety and on-line computational complexity points of views, Equation (25) together with a pre-specified threshold value, λ, is a possible candidate for the on-line fault detection method. By choosing the threshold value carefully based upon the modeling uncertainty and possible measurement noises, we will have an effective on-line fault detection mechanism. Under this conservative fault detection method, miss detection becomes trivial since the control objective is to keep the tracking error as small as possible within an affordable control effort and if the fault cannot be seen on the tracking error or it lasts only a short transient period such that the failure alarm is not triggered, the fault is not within our concern (i.e., its effect on the system performance does not degrade the control performance.) Of course, the price of the trivial miss detection and computational simplicity is the increasing possibility of false alarm possibly caused by unexpected interferences or noises.

4.4 On-line estimation of the failure dynamics

Successful on-line fault accommodation also requires a good on-line estimator to realize the numerical value of the unanticipated failure dynamics. Based upon the fact that Artificial Neural

Networks have been proven to possess the ability to approximate any piecewise continuous function given sufficient neurons in the hidden layer [23], neural networks are exploited and used as the on-line estimator for the unknown failure dynamics in this research work. Some important features of on-line learning using neural networks should first be addressed here. The structure of the on-line estimator needs to be decided (i.e., in neural networks, the number of hidden layers, number of neurons in each hidden layer, and neuron transfer functions have to be specified). It is known that neural networks are sensitive to the number of neurons in the hidden layers. Too few neurons can result in underfitting problems (poor approximation), while too many neurons may contribute to overfitting problems, where all the training patterns are well fit, but the fitting curve may take wild oscillations between the training data points [25]. The criterion for stopping the training process is another important issue in real applications. If the mean square error of the estimator is forced to reach a very small value, the estimator may perform poorly for the new input data slightly away from the training patterns. This is the well-known generalization problem. Besides, in real applications, the training patterns may contain noises since they are the measurements from real sensors. The estimator may adjust itself to fit the noise instead of the real failure dynamics. Some methods proposed to improve these problems such as early stopping criterion and generalization network training algorithms may be useful to improve these situations [24-25].

Under the on-line environment, the number of input-output data for the training purpose becomes a very important design parameter. It is apparently unreasonable and impossible to use all the measurements for training since the system dynamics may keep changing (i.e., failure may be time-varying) and the estimator may be mislead by invalid old data. A reasonable way is to use the most recent input-output measurements. A set, B, that contains the most recent measurements within a fixed length of a time-shifting data window is used to collect the training patterns,

$$B = \left\{ \left(\underline{p}(m), \underline{t}(m) \right) \middle| \underline{p} \in \Re^{S}; \underline{t} \in \Re^{T}; k - j + 1 \leq m \leq k \right\}, \tag{26}$$

where $\underline{p}(m)$ and $\underline{t}(m)$ are the network input vector and desired output vector at time step m, respectively. k is the current time step and j represents the length of the time-shifting data window which is a design parameter. This parameter has to be decided based upon the system computational capability, sampling rate, and the performance requirement. In addition, the maximum number of the effective control signal searching iterations is another important design parameter in real-time applications. It has to be within an allowable range according to the system computational capacity in the on-line situation. For gradient descent type of optimization algorithms, a time-varying learning rate can be used to possibly reduce the searching time.

5. SIMULATION STUDY

5.1 A benchmark problem: three-tank system

A well-regarded fault diagnosis benchmark shown in Figure 3 [26], the three-tank system, is used to demonstrate how to implement the proposed FTC technique in a real application. The state equations of the system are given as

$$\dot{x}_1 = (-c_1 S_p sign(x_1 - x_3)\sqrt{2g|x_1 - x_3|} + u_1)/A + \eta_1(x,u), \tag{27a}$$

$$\dot{x}_2 = (-c_3 S_p sign(x_2 - x_3)\sqrt{2g|x_2 - x_3|} - c_2 S_p \sqrt{2gx_2} + u_2)/A - q_{20} + \eta_2(x,u), \tag{27b}$$

$$\dot{x}_3 = (c_1 S_p sign(x_1 - x_3)\sqrt{2g|x_1 - x_3|} - c_3 S_p sign(x_3 - x_2)\sqrt{2g|x_3 - x_2|})/A + \eta_3(x,u). \tag{27c}$$

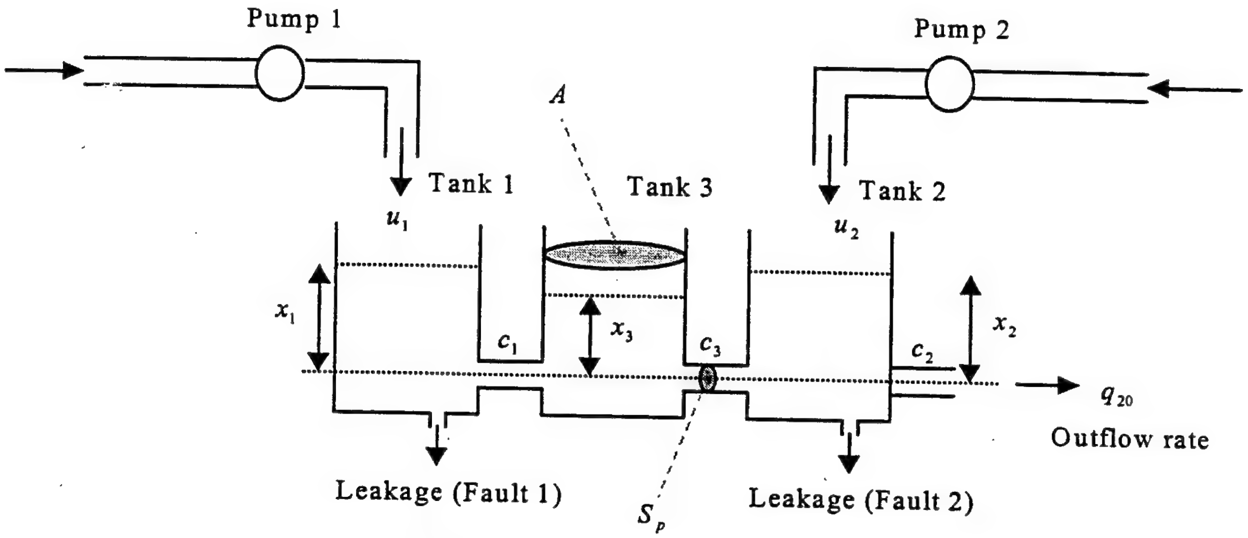


Figure 3 A benchmark problem (three-tank system)

Three tanks are identical and have a cylindrical shape with cross section $A = 0.0154\ m^2$. The cross section of the connection pipes is $S_p = 5 \cdot 10^{-5}\ m^2$ and the liquid levels in the three tanks are denoted by x_1, x_2, and x_3, respectively with $(0 \le x_i \le 0.69\ m, \forall i = 1,2,3)$. The control inputs, u_1 and u_2, are the flow rates coming from pumps 1 and 2 to the tanks 1 and 2, respectively. $q_{20} = c_1 S_p \sqrt{2gx_2}$ is the outflow rate from the tank 2. $c_1 = 1$, $c_2 = 0.8$, and $c_3 = 1$ denote the non-dimensional outflow coefficients, g is the gravity acceleration, and $\eta_i, i = 1, 2, 3$ represent the corresponding modeling uncertainty due to the inaccuracy on the cross section of connection pipes. The discrete-time model is derived by using forward Euler approximation

$$\dot{x}_i \approx \frac{x_i(k+1) - x_i(k)}{\Delta t}, i = 1,2,3, \tag{28}$$

where $\Delta t = 0.1$ second represents the sampling period. Plugging in Equation (28) and re-arranging the state equations, we have the nominal model in the controllability canonical form. Initial condition is set to be the liquid levels $x_1(0) = x_2(0) = x_3(0) = 0.15\ m$ and the control objective is to keep the liquid levels at $0.2\ m$ (i.e., $x_{1d}(k) = x_{2d}(k) = x_{3d}(k) = 0.2, \forall k > 0$). The modeling uncertainty is assumed to satisfy:

$$|\eta_i(x,u)| \le \bar{\eta}_i, \forall (x,u) \in \Pi,\ i = 1, 2, 3, \tag{29}$$

where Π represents the region of interest. In order to simulate the effects of modeling uncertainty and possibly noises in the measurements, uniformly distributed random values satisfying Equation (29) with $\bar{\eta}_1 = 3.5\times10^{-4}, \bar{\eta}_2 = 2.05\times10^{-4}$, and $\bar{\eta}_3 = 6.5\times10^{-5}$ are added to the corresponding state equations.

Design of nominal controllers

The design of the nominal controller is based upon Equation (18) without the on-line estimator and only the liquid levels $x_1(k)$ and $x_2(k)$ need to be considered in the controller design process since $x_3(k)$ will eventually reach the same level as long as we can keep $x_1(k) = x_2(k) = 0.2\ m$ due to the U tube principle. Thus, the nominal controllers are

$$u_1(k) = [\bar{Y}1(k)(a+\frac{1}{\Delta t})^{-1} - f1(\cdot)]/g1(\cdot), \tag{30a}$$

$$u_2(k) = [\bar{Y}2(k)(a+\frac{1}{\Delta t})^{-1} - f2(\cdot)]/g2(\cdot), \tag{30b}$$

where

$$\bar{Y}1(k) = \frac{x_{1d}(k+1) - x_{1d}(k) + x_1(k)}{\Delta t} + ax_{1d}(k+1),$$

$$\bar{Y}2(k) = \frac{x_{2d}(k+1) - x_{2d}(k) + x_2(k)}{\Delta t} + ax_{2d}(k+1).$$

$f1(\cdot)$, $f2(\cdot)$, $g1(\cdot)$, and $g2(\cdot)$ are the corresponding terms obtained when we re-organized the nominal model into the controllability canonical form. The two S functions, S_1 and S_2, are defined for x_1 and x_2, respectively, with the same form as Equation (4) and $a = 10$. The sum of the mean square errors (i.e., $\tilde{x}_1^{\,2}(k) = [x_{1d}(k) - x_1(k)]^2$ and $\tilde{x}_2^{\,2}(k) = [x_{2d}(k) - x_2(k)]^2$) within a fixed length (i.e., 5 time steps) of time-shifting window is selected as a criterion to test performance of the nominal controllers with the presences of modeling uncertainty and possible noises. Based upon the testing results, the fault detection threshold value, λ, is then selected as 2.0×10^{-3} in steady state condition.

Multiple failures: leakages in the tanks

Consider an abrupt leakage in the tank 1 and an incipient leakage in the tank 2 whose failure dynamics are

$$F_1(k) = -c_1\pi r_1^2\sqrt{2gx_1(k)},\ \beta_1(k-T_1) = U(k-T_1),\ T_1 = 270, \tag{31a}$$

$$F_2(k) = -c_2\pi r_2^2\sqrt{2gx_2(k)}, \beta_2(k-T_2) = (1-e^{-\alpha_2(k-T_2)})U(k-T_2),\ \alpha_2 = 0.063, T_2 = 426, \tag{31b}$$

where $r_1 = 7.3\times10^{-2}$ and $r_2 = 8.4\times10^{-2}$. No information in Equation (31) is assumed to be known except that the state variables $x_1(k)$ and $x_2(k)$ are measurable. The physical knowledge of the system provides us useful information to determine the initial upper bound (η_{old}) of the failure dynamics. Since the failures are possible leakage problems in the tanks (i.e., the failures of system components), the maximum effect caused by the failure is sudden draining of the liquid in the tank, which corresponds to the worst failure condition where the tank is completely broken. Thus, the initial upper bounds for failures can be chosen as the liquid levels in the tanks at the corresponding time step. Two separate multi-layer perceptron network (MLP) with single input neuron, five neurons in the first hidden layer, five neurons in the second hidden layer, and single output neuron (1-5-5-1) are used to serve as the on-line failure estimators for F_1 and F_2, respectively, with the static backpropagation method as the training algorithm. The selection of the MLP network structure is a design parameter and may not be optimal in this case. Generally speaking, a more complicated structure may be required for a more complex function to achieve a better performance and the computational cost is expected to increase with the complexity. The on-line approximation result is monitored by the criterion shown in Equation (24) with $l = 10$ and the least upper bound of the failure uncertainty is computed according to Equation (22). Figure 4 shows the liquid levels in the tanks under the nominal control law alone. As the first leakage in tank 1 occurs, the liquid level 1 drops quickly causing dropping of the liquid level in the tank 3. As the second leakage problem occurs in the tank 2, the liquid levels eventually drop below the initial condition. Applying the proposed control technique with the corrective control law (Equation (19)), we observe significant performance improvement by proper reconfiguration of the control inputs, flow rates from pump 1 and 2, as shown in Figure 5. The implementation of the first control law (Equation (18)) for this problem is straightforward and the result is shown in Figure 6. (i.e., with the same MLP estimators and the delay of compensation.)

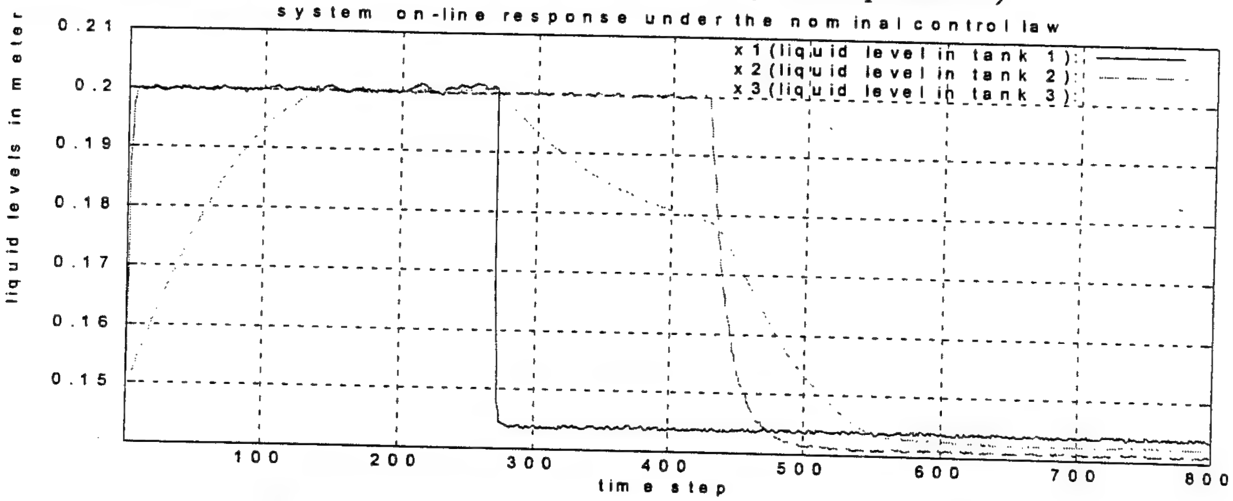


Figure 4 System on-line response with the nominal control law

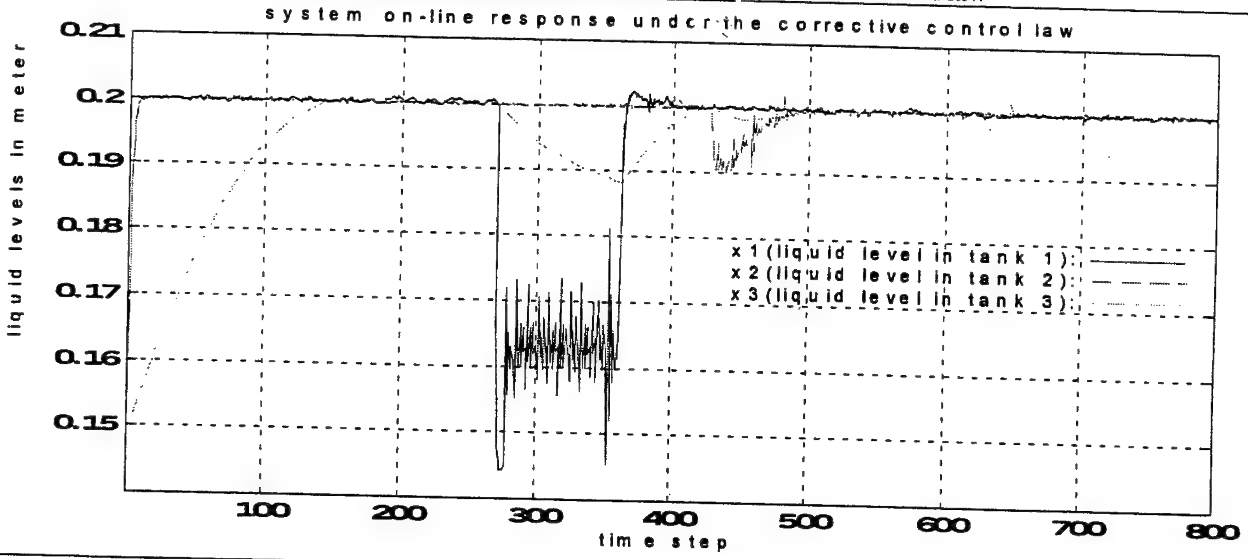


Figure 5 System on-line response with the alternative corrective control law

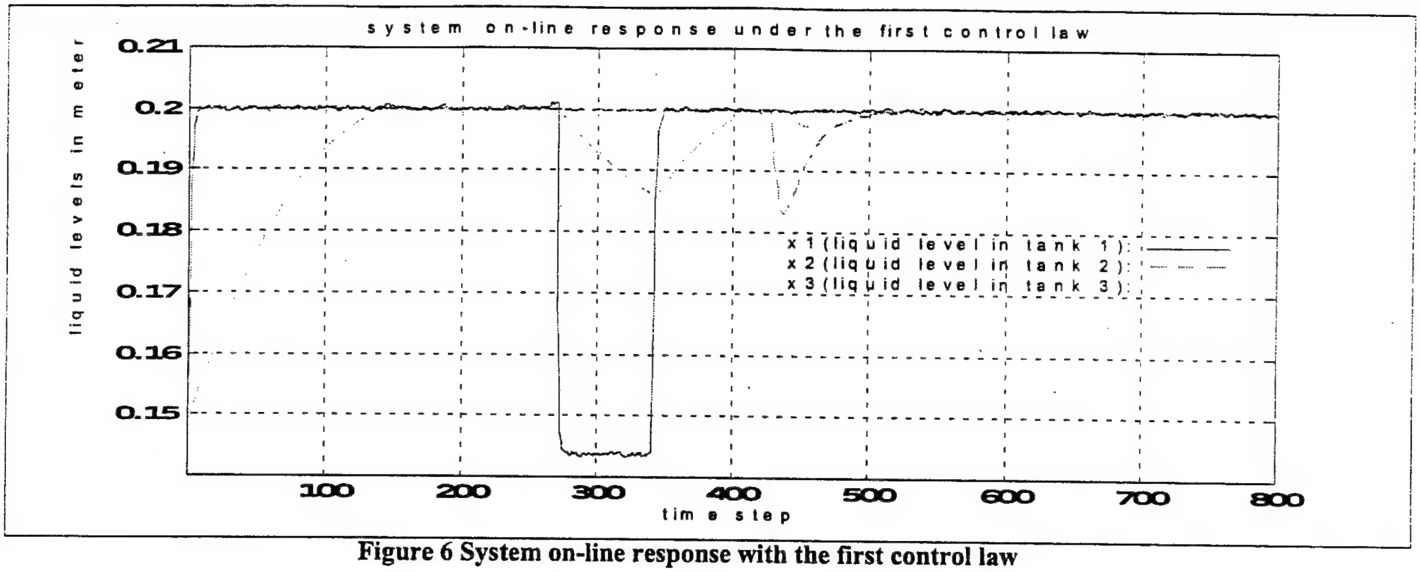


Figure 6 System on-line response with the first control law

5.2 On-line fault accommodation for multiple time-varying failures

In order to demonstrate how to use the proposed FTC strategy for the general nonlinear system under the general failure scenario, a MIMO system under multiple time-varying failures is considered as shown in Equation (32),

$$\begin{aligned} y_1(k+1) &= \frac{y_1(k)}{1+y_2(k)^2} + u_1(k)^2 + u_2(k)^2 - 16u_1(k) - 20u_2(k) + \Delta f_1(k), \\ y_2(k+1) &= \frac{y_1(k)y_2(k)}{1+y_2(k)^2} + u_1(k)u_2(k) + 20u_1(k) - 5u_2(k) + \Delta f_2(k), \\ \Delta f_1(k) &= \beta_{10}(k-T_{10}) \times 0.1 \times \frac{k-25}{20} y_1(k)\cos(u_1(k)) + \beta_{11}(k-T_{11}) \times 0.6 y_1(k) y_2(k), \\ \Delta f_2(k) &= \beta_{20}(k-T_{20}) \times 0.1 \times y_1(k)y_2(k), \end{aligned} \tag{32}$$

where $\beta_{10}(k-T_{10}) = U(k-T_{10})$, $\beta_{11}(k-T_{11}) = U(k-T_{11})$, $\beta_{20}(k-T_{20}) = U(k-T_{20})$, $T_{10} = 25$, $T_{20} = 15$, and $T_{11} = 123$. The nominal system is first realized through a 4-75-2 MLP neural network. 2,000 input-output training patterns are collected by supplying uniformly distributed random inputs varying from –1.5 to 1.5. A 3-4-2 MLP network is chosen as the on-line estimator and the Levenberg-Marquardt with Bayesian regularization algorithm [24-25] is used in the training process for both the NN nominal model and the NN on-line estimator (i.e., a 4-40-2 MLP network is used as a nominal controller trained off-line by following the design procedure shown in Section 4.2). The desired trajectories for outputs y_1 and y_2 are generated by Equation (33)

$$
\begin{aligned}
&ref_1(k) = 0.2\times\sin(\frac{k\pi}{100}),\\
&y_{1d}(k+1) = 0.6y_{1d}(k) + 0.2y_{1d}(k-1) + ref_1(k),\\
&ref_2(k) = 0.2\times\sin(\frac{k\pi}{75}),\\
&y_{2d}(k+1) = 0.3y_{2d}(k) + 0.2y_{2d}(k-1) + ref_2(k).
\end{aligned}
\tag{33}
$$

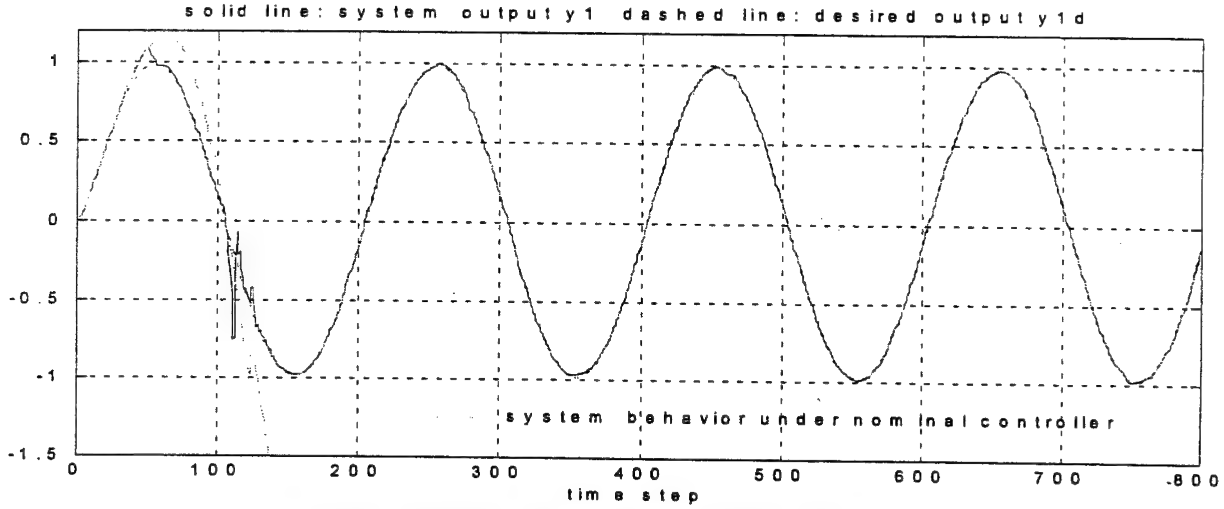


Figure 7 System response, y1, vs. desired output, y1d (MIMO system; 25-data window; consecutive abrupt faults)

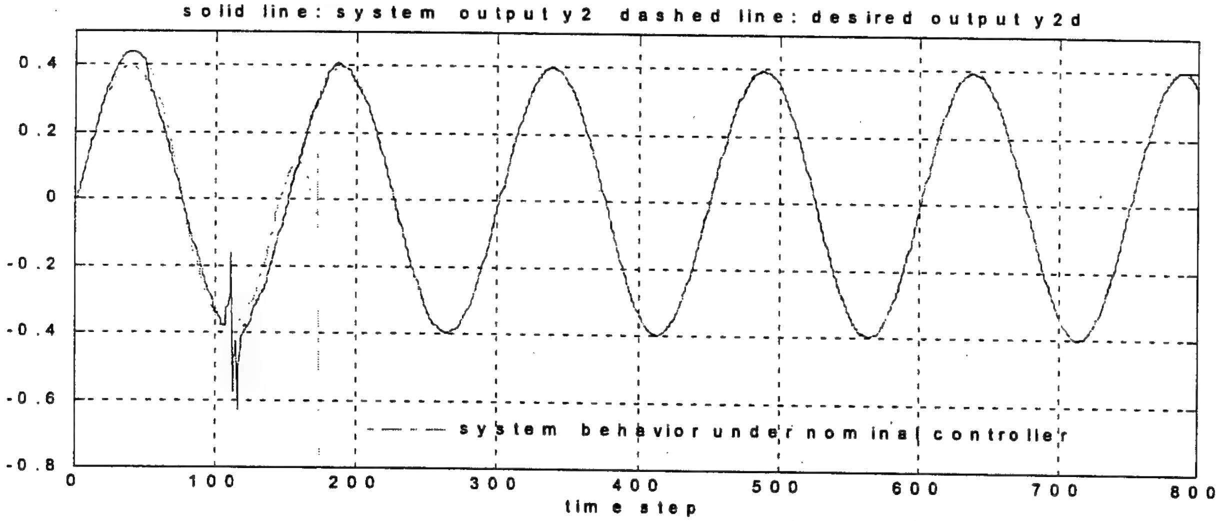


Figure 8 System response, y2, vs. desired output, y2d (MIMO system; 25-data window; consecutive abrupt faults)

The testing criterion for the nominal controller is similar to Equation (25) with the sum of the mean square error for both outputs and the length of evaluation is selected as 5. Based upon the testing result, the threshold value of the on-line fault detection scheme is chosen as 9×10^{-5} under noise-free situations. No information pertained to the failures, Δf_1 and Δf_2, are assumed known. Two different S functions are defined for y_1 and y_2 in the same form of Equation (4) with $a = 10$, respectively. A simple mean value of the two estimated gradient directions realized through the NN on-line estimator is used for the searching of the effective control inputs. The length of the time-shifting data window, j in Equation (26), is selected as 25. The on-line simulation speed can reach 2-3 time steps per second under Intel Pentium-II 450 dual processors. Figures 7 and 8 show the system response for the first output and the response for the second output together with the desired outputs, respectively, while the nominal controller alone fails to maintain the system stability under multiple failures.

6. Real-Time Experiment

To obtain a comprehensive insight for quantification of the design parameters and the real-time control system, an on-line fault tolerant control test bed to validate the proposed on-line fault tolerant control strategy in real hardware has been constructed. The hardware setup is shown in Figure 9. It consists of the following major components,

1. a BALDOR dc motor with maximum ½ hp,
2. a MAGTROL HD-505-8N dynamometer,
3. a MAGTROL 6200 dynamometer controller/readout,
4. one ADVANCED dc motor amplifier,
5. dSPACE software, DS1102 board and cable box with Texas Instruments TMS320C31 floating-point Digital Signal Processor (DSP), and
6. NT workstation with Intel Pentium II-450 dual processors.

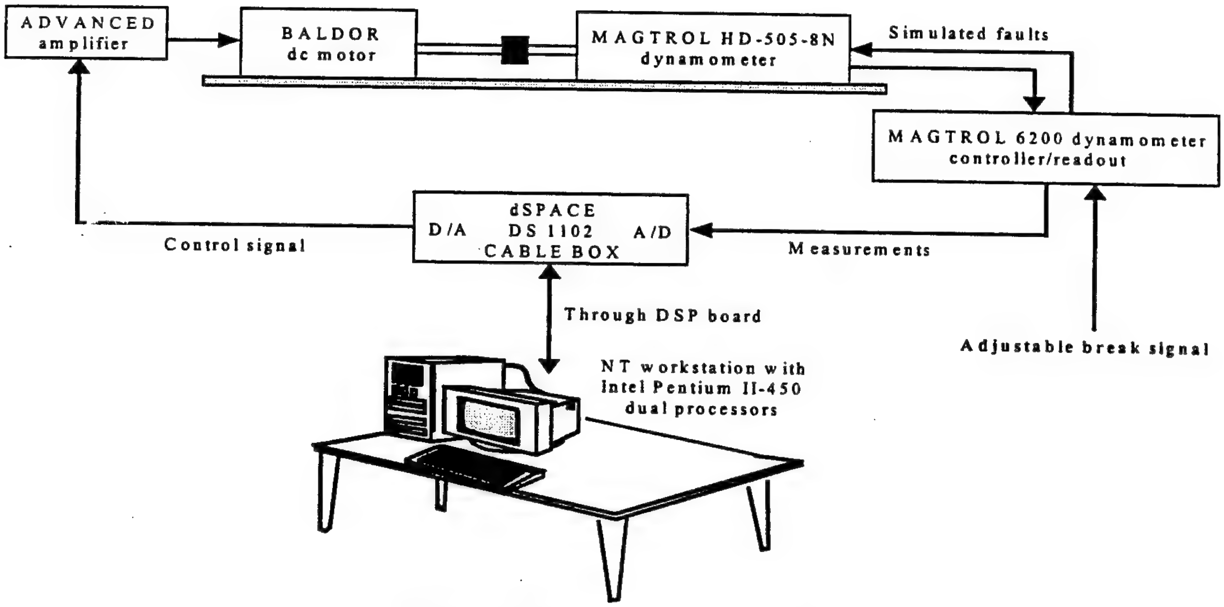


Figure 9 Hardware experiment setup

The dc motor is connected to the dynamometer that is used to generate unanticipated friction on the motor shaft to simulate the unanticipated system failures. The control objective is to maintain the rotational speed of the motor (i.e., in terms of rpm) to the desired patterns with the presence of the unanticipated simulated failures. A computer with Intel Pentium II-450 dual processors is used to simulate the intelligent control regulator, fault detection mechanism, and on-line estimator. An embedded encoder and sensor in the dynamometer provide motor rpm and torque measurements in real-time, respectively. The measured signals are connected to a MAGTROL 6200 dynamometer controller/readout with the on-line readings shown on the device screen and the same signals are sent to a dSPACE DS1102 cable box that is connected to the workstation through the TMS320C31 DSP board. An adjustable break dial on the front panel of the dynamometer controller is used to generate the simulated time-varying, unknown and unanticipated breaks (i.e., unanticipated workload on the dc motor). All the necessary computation and the appropriate control input to drive the motor are computed within the workstation. The DSP board and dSPACE software are used to provide the necessary interface (A/D and D/A converter) and the integration of the real-time control with high-level languages including MATLAB, SIMULINK, and C programs. A picture of the real-time fault tolerant control test bed is shown in Figure 10.

In real-time environment, to close the on-line control loop as shown in Figure 9, an application source code (i.e., obj file) has to be created and downloaded to the TMS320C31 DSP. The on-line fault detection scheme, failure estimation, and control algorithm are performed under Matlab workspace in the NT workstation which communicates with the DSP through dSPACE MLIB (Matlab-dSPACE Interface Library). Figure 11 shows the SIMULINK model that is used to create the application source code for the real-time experiment. One 14-bit D/A converter channel and one 16-bit A/D converter channel are used to generate the control input (i.e., motor input voltage) and collect the torque reading from the dynamometer controller/readout, respectively. A discrete filter is used to reduce the effect of measurement noises in the torque reading. The rotational speed reading is decoded through one DS1102 encoder interface channel with a 24-bit counter. The DSP with generated application code runs the hardware experiments in real-time with sampling period 0.01 second and the control signal generated by the computer is sent to regulate the real-time response by changing the constant value in the SIMULINK model.

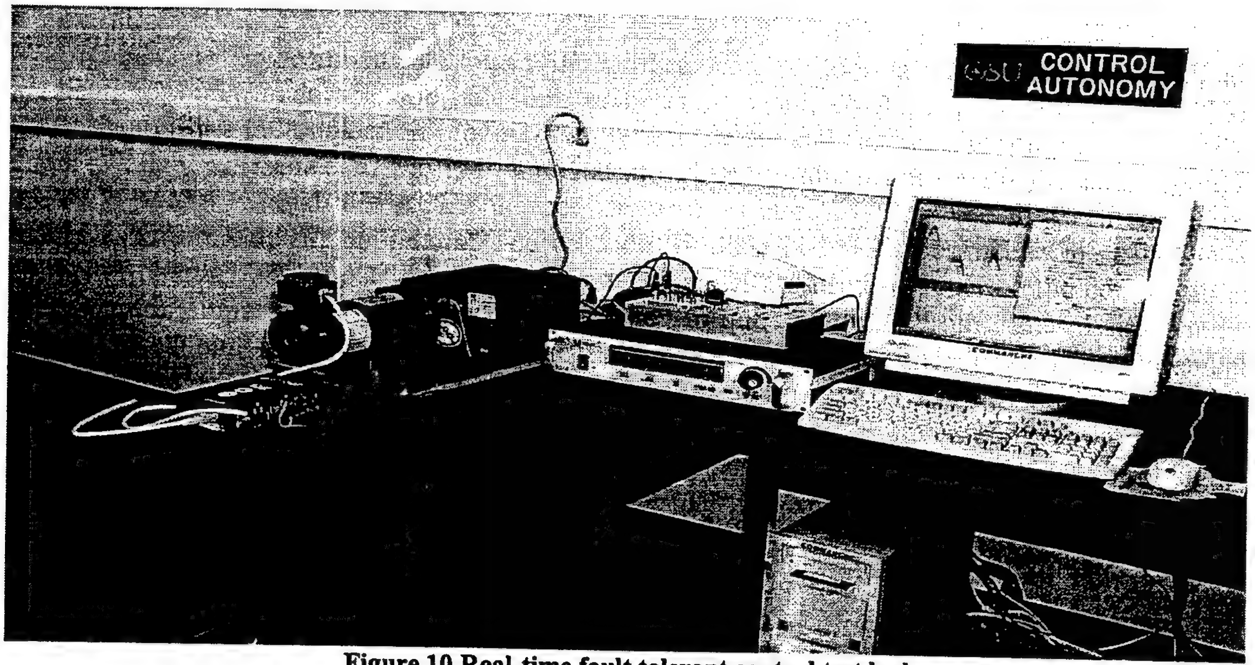


Figure 10 Real-time fault tolerant control test bed

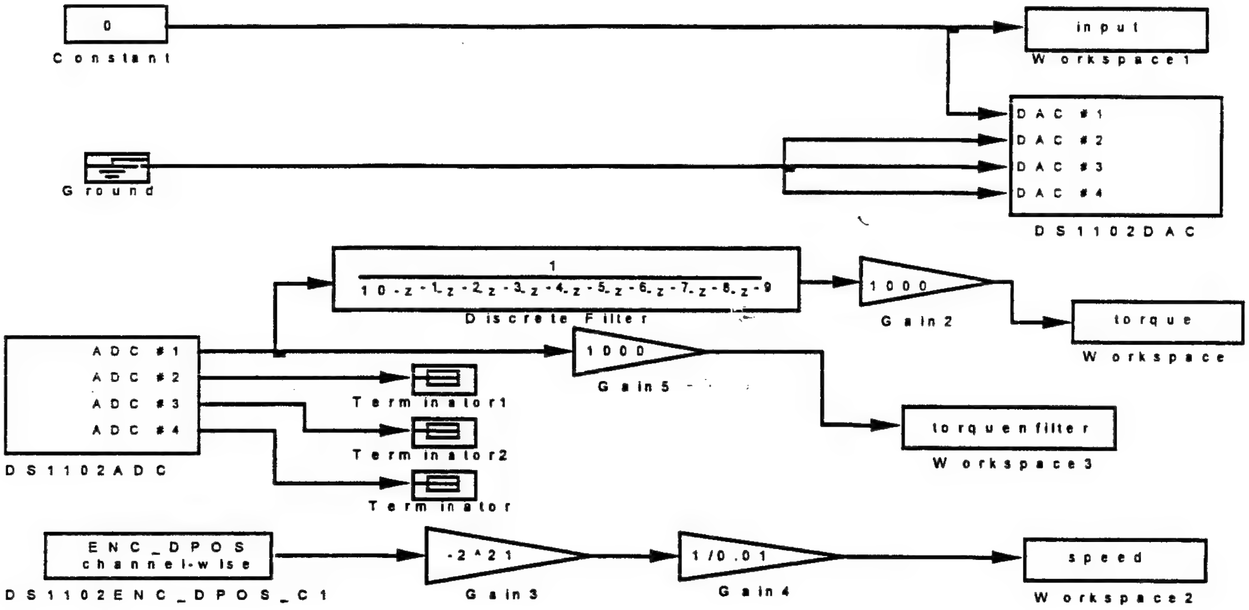


Figure 11 The SIMULINK model for the real-time experiment

Following the design procedure shown in Section 4, the first step is to obtain a nominal model for the fault-free system. It is well known that a dc motor can be modeled as a linear time-invariant system. For an armature-controlled dc motor with the negligible time constant of the armature, the nominal transfer function can be represented by Equation (34) [27],

$$G(s)=\frac{w(s)}{V(s)}=\frac{K_m}{[R_a(Js+f)+K_bK_m]}, \tag{34}$$

where $w(s)$ and $V(s)$ denote the rotational speed and motor voltage in s domain, respectively. K_b, K_m, R_a, J, and f are motor constants. Equation (34) can be re-organized as Equation (35) with A, b, c representing the corresponding constants,

$$A\dot{w}+bw=cV. \tag{35}$$

Using the forward Euler approximation shown in Equation (28), the discrete-time nominal model can be derived and shown in Equation (36),

$$\begin{aligned} w(k+1) &= [1-b\Delta t/A]w(k)+[c\Delta t/A]V(k) \\ &= f_{linear}w(k)+g_{linear}V(k), \end{aligned} \tag{36}$$

which is in the controllability canonical form. The next step is to identify the parameters, $f_{linear}=[1-b\Delta t/A]$ and $g_{linear}=[c\Delta t/A]$. Since Equation (36) is a linear time-invariant system, the batch form least square estimation method can be used for the identification of the parameters [28]. With the zero initial condition, 20,000 sets of input signals generated by Equation (37) are sent to the system for the collection of the system responses, $w(k)$,

$$V(k) = 0.015 \times \sin(\frac{k\pi}{2000}) + 0.015 \,,\; k = 1,2,\ldots,19{,}999. \tag{37}$$

The batch form least square method provides the parameter estimation as follows [28],

$$Z = \begin{bmatrix} w(20{,}000) \\ w(19{,}999) \\ \vdots \\ w(2) \end{bmatrix}, H = \begin{bmatrix} w(19{,}999) & V(19{,}999) \\ w(19{,}998) & V(19{,}998) \\ \vdots & \vdots \\ w(1) & V(1) \end{bmatrix}, \theta = \begin{bmatrix} f_{linear} \\ g_{linear} \end{bmatrix}, \; Z = H\theta + v \text{, and}$$

$$\hat{\theta}_{LS} = \begin{bmatrix} \hat{f}_{linear} \\ \hat{g}_{linear} \end{bmatrix} = [H^T H]^{-1} H^T Z \,, \tag{38}$$

where v and $\hat{\theta}_{LS}$ represent the white noise and the least square estimation, respectively. The design of the nominal controller follows Equation (18) without the term of fault estimator as shown in Equation (39),

$$\bar{Y}(k) = \frac{w_{desired}(k+1) - w_{desired}(k) - w(k)}{\Delta t} + a w_{desired}(k+1) \text{ and}$$

$$V_{no\min al}(k) = [\bar{Y}(k)(a + \frac{1}{\Delta t})^{-1} - \hat{f}_{linear} \times w(k)] \frac{1}{\hat{g}_{linear}} \tag{39}$$

with $a = 1$ and the S function defined in the form of Equation (4).

The final step in the off-line design stage is to evaluate the nominal model accuracy and the performance of the nominal controller under the fault-free environment for proper selection of the design parameters in the on-line fault detection scheme. The length of the time-shifting evaluation window for the fault detection scheme (i.e., Equation (25) with the square operation replaced by the absolute value) is pre-selected as 5 and the system response under the nominal controller is tested using this criterion with the presence of measurement noises. The on-line fault detection threshold value is decided as 100 based upon the testing results. Under the unanticipated failures, the system response can be approximated by Equation (40),

$$w(k+1) = f_{linear} w(k) + g_{linear} V(k) + F(Torque(k)) \,, \tag{40}$$

where F denotes the unknown effect that changes the motor rotational speed due to the unanticipated workload, $Torque(k)$. To reduce the negative effect of noisy measurements, the approximation target (i.e., numerical value of F) is computed based upon the average of the differences between the nominal model outputs and the actual speed readings every 10 time steps. A 1-5-5-1 MLP network is used to approximate the unknown failure effect, F, on-line with the static backpropagation algorithm. Two real-time experiments with different desired trajectories and unanticipated faults are presented here to test the proposed fault accommodation technique. Each real-time experiment is complete within 5,000 time steps. The design parameters of the learning result criterion (i.e., Equations (19)-(24)) for the alternative corrective control law are $l = 20$ and $\delta = 10$.

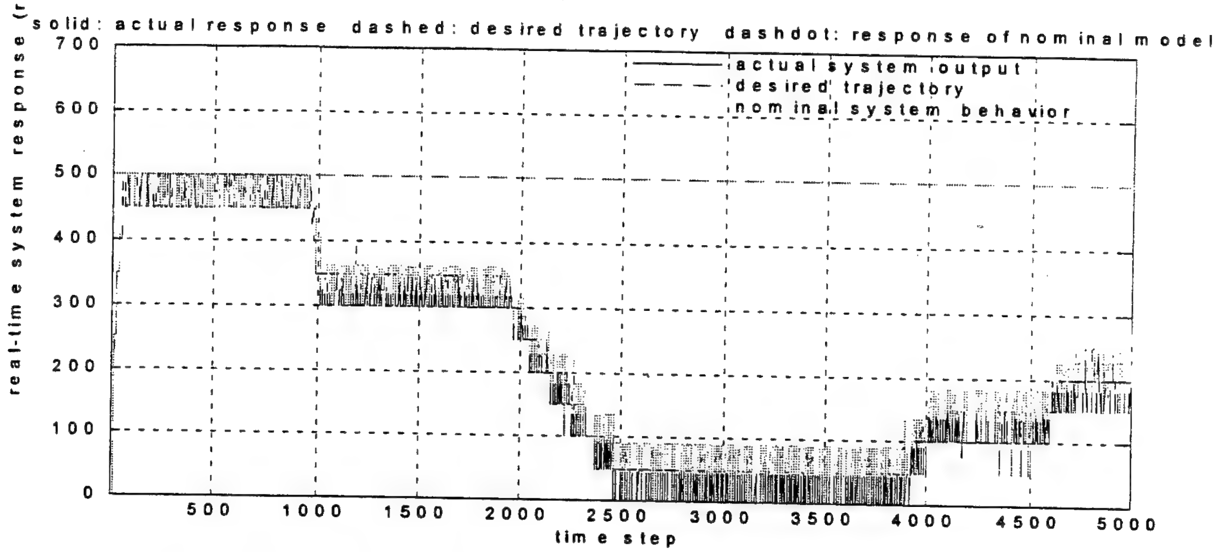


Figure 12 On-line system behavior under nominal controller only (experiment 1)

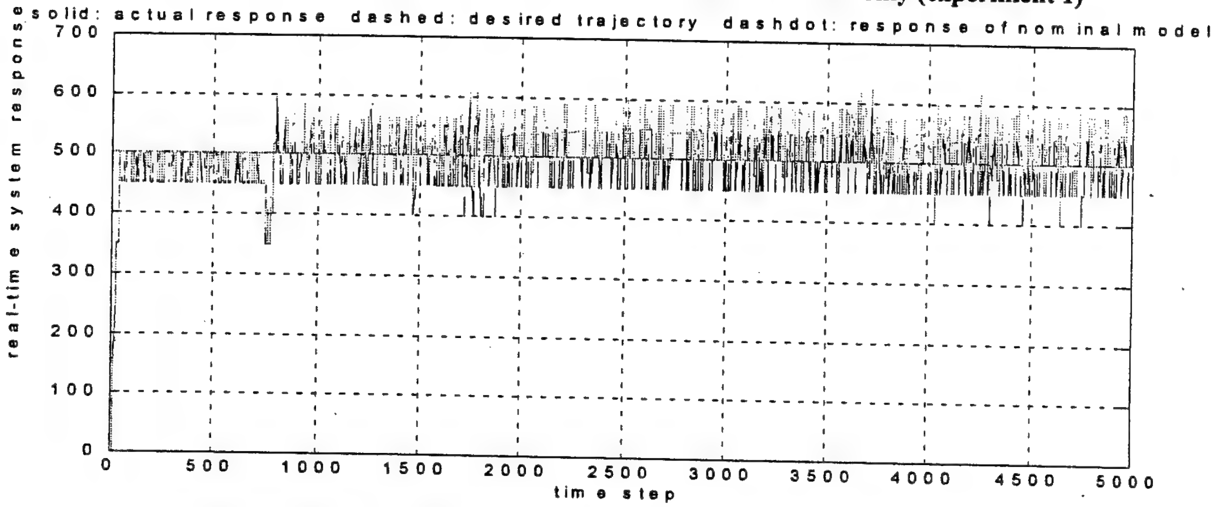


Figure 13 On-line system behavior under the first control law (experiment 1)

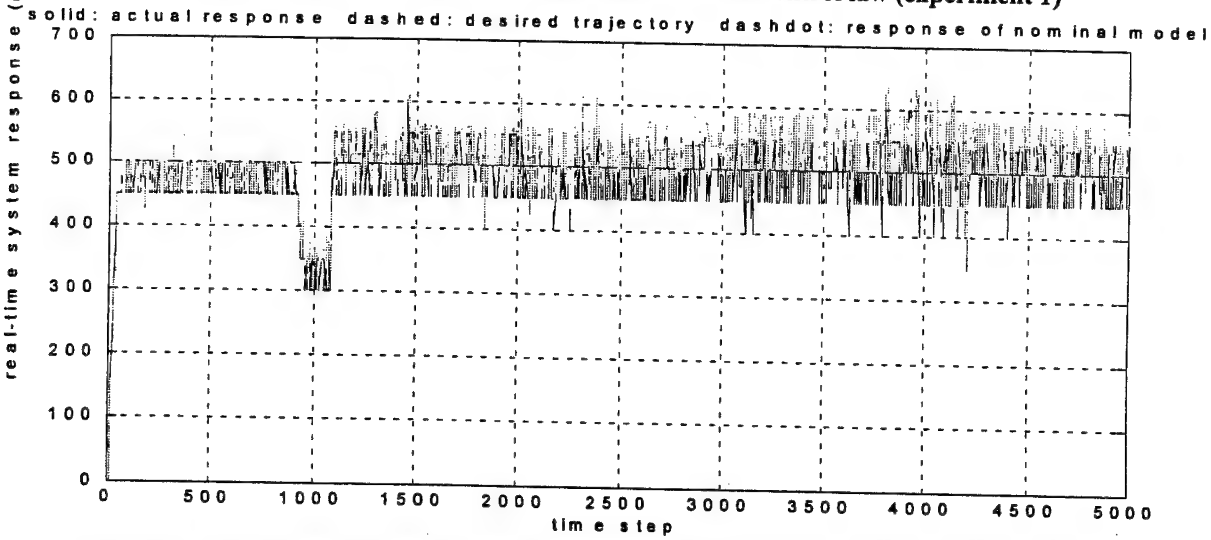


Figure 14 On-line system behavior under the alternative corrective control law (experiment 1)

6.1 Experiment 1

The control objective in this experiment is to maintain constant rotational speed at 500 rpm. The unknown and unanticipated failure is generated by on-line adjusting the break dial on the front panel of the dynamometer controller/readout. The actual time-varying workload on the motor is unknown with the torque reading 18-30% shown on the front panel of the dynamometer controller/readout while the break is hand adjusting during the real-time experiment. Figure 12 is the real-time system behavior plot under the nominal controller. Due to the increased workload, the motor rotational speed almost reaches zero from time step 2500 to 3850. On the other hand, successful fault tolerant mission has been accomplished through the proposed on-line fault accommodation technique as shown in Figures 13 and 14.

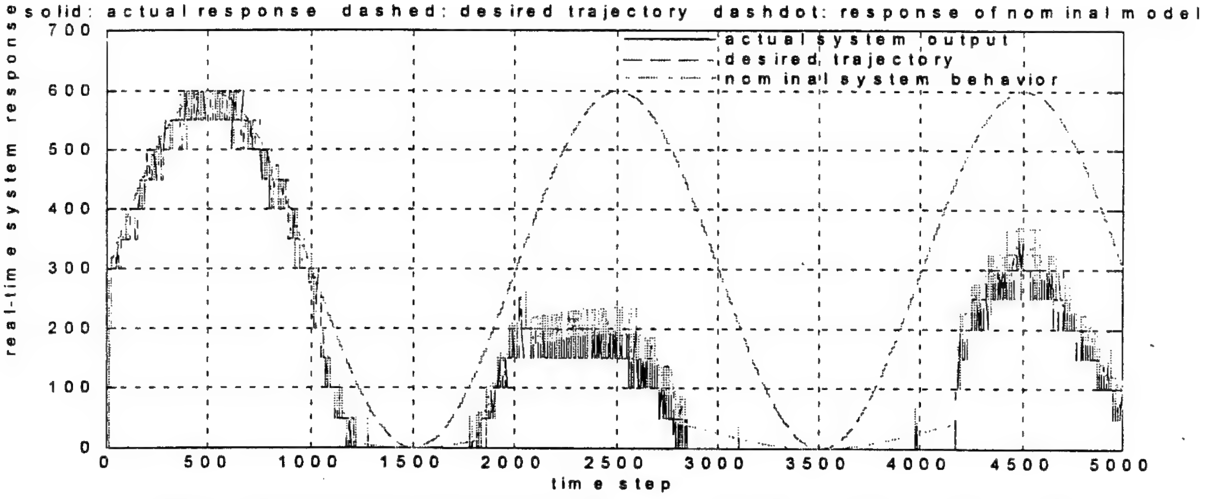


Figure 15 On-line system behavior under nominal controller only (experiment 2)

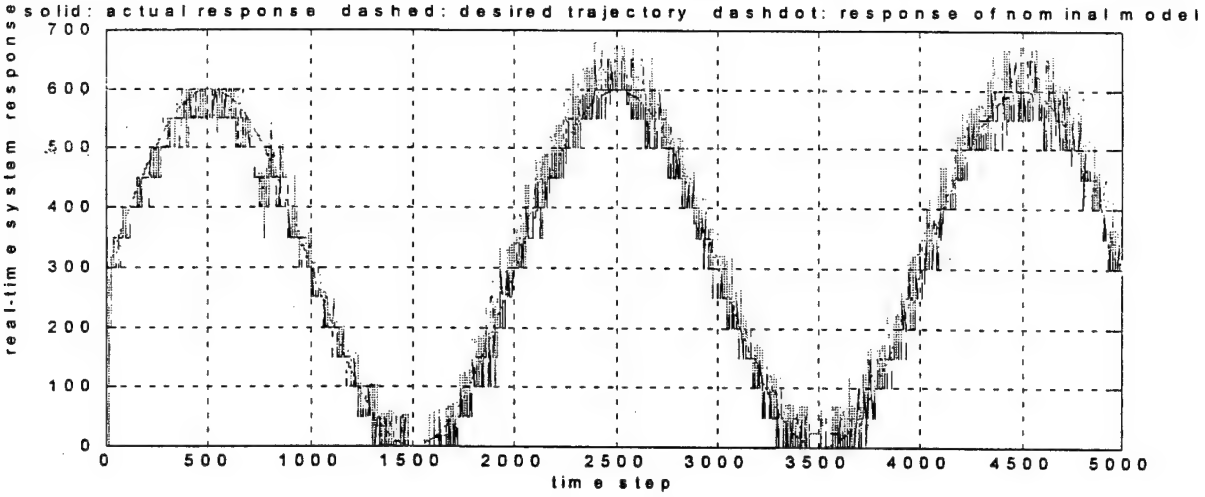


Figure 16 On-line system behavior under the first control law (experiment 2)

6.2 Experiment 2

The desired trajectory in this experiment is selected as a sinusoid curve generated by a linear model with a reference input as shown in Equation (41),

$$\begin{aligned} ref(k) &= 60\times\sin(\frac{k\pi}{1000})+60, \\ w_{desired}(k+1) &= 0.6w_{desired}(k)+0.2w_{desired}(k-1)+ref(k). \end{aligned} \tag{41}$$

The unknown workload used to generate the simulated unanticipated faults ranges from 15% to 28%. The system behavior under the failures with the nominal controller alone is plotted in Figure 15. As clearly shown, the performance has been significantly degraded and the rotation actually stops during the time periods, from time step 1300 to 1700 and 3200 to 4000, due to the relatively large unanticipated workload. Figures 16 and 17 show the satisfactory real-time fault accommodation when the first and the alternative corrective control techniques are applied, respectively.

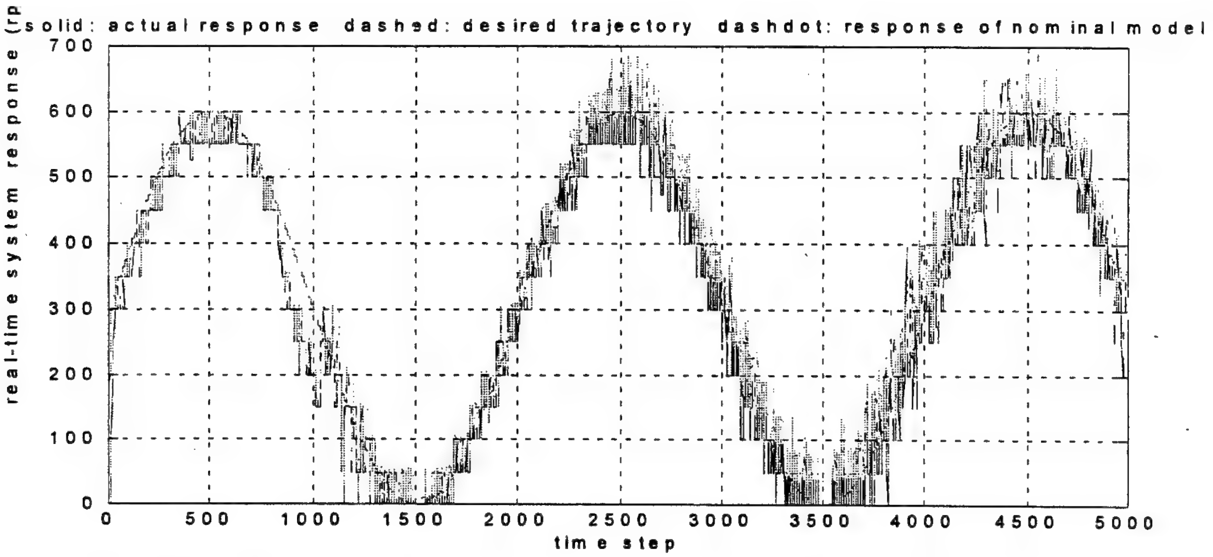


Figure 17 On-line system behavior under the alternative corrective control law (experiment 2)

7. Conclusion

In this paper, the on-line fault tolerant control problems under unanticipated system failures are investigated form a realistic point of view. Through the discrete-time Laypunov stability theory, the necessary and sufficient conditions to guarantee the system on-line stability and performance under failures are derived under no specific assumption of the system dynamic structure and failure scenarios. An on-line fault accommodation control strategy that contains detail off-line design procedure, an efficient on-line fault detection scheme, and effective control law reconfiguration technique is presented for the FTC problems of interest. Because of its capabilities of *self-optimization* and *on-line adaptation*, Artificial Neural Network is used in this research work as the on-line estimator for the unknown failure dynamics.

The effective control inputs to accommodate system failures are automatically computed on-line by the control regulator through the realization of the NN estimator based upon only partial available information of the failure dynamics. The price paid for this achievement of the successful control mission relies on a certain degree of computational expense. Under the suggested on-line fault detection scheme, the miss detection of failures becomes trivial while the possibility of false alarm situations increases. Due to space limitation, the simulation tests under noisy measurements and false alarms are omitted. However, it is recommended that a noise reduction or cancellation process be used for better system performance in real applications, if any prior information concerning the statistical property of the noise is available since the contaminated noisy measurements will mislead the interpretation of the system behaviors and the on-line estimator. In general, the effectiveness of the developed on-line fault accommodation control technique for catastrophic system failures has been validated through on-line simulation tests. The successful on-line fault tolerance in real applications has also been demonstrated through real-time hardware experiments with the presences of measurement noises and unknown unanticipated faults. The on-line simulation speed can reach 2-3 time steps per second under the Intel Pentium-II 450 dual processors. Real-time experimental results also indicate that a more powerful computing device such as a computer with higher speed dual processors is mandatory for the on-line real-time fault tolerant control in the real applications under the more general formulations. Although the currently used dual processors may not be fast enough in many real-time control systems that require higher sampling rate, it is believed that with the continuous performance improvement of microprocessors and semiconductor technology, the developed on-line fault accommodation technique can be implemented on-line in most of the real-time control systems in near future.

Current research work is focusing on extending the on-line fault tolerant control technique under conflicting failure situations in MIMO cases, where the accommodations of some failures may require a certain degree of compromise in other objectives. Under these situations, the control problems are both theoretically and technically complicated since the reconfiguration of the control actions becomes a multi-objective optimization in the sense of Pareto optimality.

REFERENCES

[1] Z. Gao, and P. Antsaklis, "Reconfigurable control system and design via perfect model following," *International Journal of Control*, Vol. 56, No. 4, pp. 783-798, 1992.

[2] J. Jiang, "Design of reconfigurable control systems using eigenstructure assignment," *International Journal of Control*, Vol. 59, No. 2, pp. 395-410, 1994.

[3] D. Sauter, F. Hamelin, and H. Noura, "Fault tolerant control in dynamic systems using convex optimization," *Proceedings of the IEEE International Symposium on Intelligent Control*, pp. 187-192, 1998.

[4] D. Theilliol, H. Noura, and D. Sauter, "Fault-tolerant control method for actuator and component faults," *Proceedings of the IEEE Conference on Decision and Control*, pp. 604-609, 1998.

[5] M. Bodson, and J. Groszkiewicz, "Multivariable adaptive algorithms for reconfigurable flight control," *IEEE Transactions on Control Systems Technology*, Vol. 5, No. 2, pp. 217-229, 1997.

[6] J. Jiang, and Q. Zhao, "Fault tolerant control system synthesis using imprecise fault identification and reconfigurable control," *Proceedings of the IEEE International Symposium on Intelligent Control*, pp. 169-174, 1998.

[7] M. Polycarpou, and A. Vemuri, "Learning methodology for failure detection and accommodation," *IEEE Control Systems Magazine*, Vol. 15, No. 3, pp. 16-24, 1995.

[8] M. Polycarpou, "Stable learning scheme for failure detection and accommodation," *Proceedings of the IEEE International Symposium on Intelligent Control*, pp. 315-320, 1994.

[9] K. Narendra, and S. Mukhopadhyay, "Adaptive control using neural networks and approximate models," *IEEE Transactions on Neural Networks*, Vol. 8, No. 3. pp. 475-485, 1997.

[10] J. Michael, and R Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Computation*, Vol. 6, No. 2, pp 181-214, 1994.

[11] C. Juang, and C. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Transactions on Fuzzy Systems*, Vol. 6, No. 1, pp. 12-32, 1998.

[12] J. Chun, M. Kim, H. Jung, and S. Hong, "Shape optimization of electromagnetic devices using immune algorithm," *IEEE Transactions on Magnetics*, Vol. 33, No. 2, pp. 1876-1879, 1997.

[13] M. Marchesi, G. Molinari, and M. Repetto, "A parallel simulated annealing algorithm for the design of magnetic structures," *IEEE Transactions on Magnetics*, Vol. 30, No. 5, pp. 3439-3442, 1994.

[14] S. Ottner, "Optimising television commercial air-time by means of a genetic algorithm," *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 761, 2000.

[15] Y. Tanaka, and T. Yoshida, "An applications of reinforcement learning to manufacturing scheduling problems," *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 4, pp. 534-539, 1999.

[16] E. Misawa, "Discrete-time sliding mode control for nonlinear systems with unmatched uncertainties and uncertain control vector," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 119, pp. 503-512, 1997.

[17] G. Yen, and L. Ho, "Fault tolerant control: an intelligent sliding model control strategy," *Proceedings of the American Control Conference*, pp. 4204-4208, 2000.

[18] L. Ho, and G. Yen, "On-line fault accommodation control for catastrophic system failures," submitted to *IEEE Transactions on Control Systems Technology*.

[19] K. Narendra, and K. Parthasathy, "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Transactions on Neural Networks*, Vol. 2, No. 2, pp. 252-262, 1991.

[20] D. Psaltis, A. Sideris, and A. Yamaura, "Neural controllers," *Proceedings of the IEEE International Conference on Neural Networks*, pp. IV 551-558, 1987.

[21] J. Spall, "Multivariable stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, Vol. 37, No. 3, pp. 332-341, 1992.

[22] M. Hagan, H. Demuth, and M. Beale, *Neural Network Design*, Boston, MA: PWS Publishing, 1996.

[23] K. Hornik, M. Stinchombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, Vol. 2, No. 5, pp. 359-366, 1989.

[24] D. Mackey, "Bayesian interpolation," *Neural Computation*, Vol. 4, No. 3, pp. 415-447, 1992.

[25] H. Demuth, and M. Beale, *Neural Network Toolbox User's Guide*, Version 3, Natick, MA: The Math Works, 1998.

[26] X. Zhang, T. Parisini, and M. Polycarpou, "Robust parametric fault detection and isolation for nonlinear systems," *Proceedings of the IEEE Conference on Decision and Control*, pp. 3102-3107, 1999.

[27] R. Dorf, and R. Bishop, *Modern Control Systems, 7th edition*, Menlo Park, CA: Addison-Wesley Publishing Co., 1995.

[28] J. Mendel, *Lessons in Estimation Theory for Signal Processing, Communications, and Control*, Englewood Cliffs, NJ: Prentice-Hall, 1995.

[29] F. Beaufays, and E. Wan, "Relating real-time backpropagation and backpropagation-through-time: an application of flow graph interreciprocity," *Neural Computation*, Vol. 6, No. 2, pp. 297-305, 1994.

[30] M. Chow, and S. Yee, "An adaptive backpropagation through time training algorithm for a neural controller," *Proceedings of the IEEE International Symposium on Intelligent Control*, pp. 170-175, 1991.

[31] P. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of IEEE*, Vol. 78, No. 10, pp. 1550-1560, 1990.

APPENDIX

Assumptions:

1. The nominal model, $N\hat{y}(k+1)$, is accurate and precise enough such that $N\tilde{y}(k+1)$, the remaining uncertainty of the nominal system, is bounded by $\sup_{\forall k>T_f}\{|N\tilde{y}(k+1)|\}$, where T_f is the starting time step that the control input starts accommodating the failures. (Note that this constraint can be possibly relaxed since the nominal model can be obtained off-line using all existing modeling techniques.)
2. The remaining uncertainty of the failure dynamics, $nf\tilde{y}(k+1)$, is the residue resulting from the difference between the actual $fy(k+1)$ and the best estimation of the on-line estimator and it is bound by the least upper bound, $\sup_{\forall k>T_f}\{|nf\tilde{y}(k+1)|\}$.
3. The error caused by the optimization algorithm is bounded by $\sup_{\forall k>T_f}\{|\Delta Error(k)|\}$.

Proof:

Let $\Delta Error(k)$ represent the error after the searching effort of the optimization algorithm. Then,

$$\Delta Error(k) = Desire(k) - N\hat{y}(k+1) - nfy(k+1).$$

By Equation (11),

$$\bar{Y}(k)(a+\frac{1}{\Delta t})^{-1} - \Delta Error(k) = N\hat{y}(k+1) + nfy(k+1). \qquad (42)$$

For $S(k) > 0$:

Plugging Equations (42) and (10) into the inequality (8), we have

$$(\bar{Y}(k)+S(k))(a+\frac{1}{\Delta t})^{-1} > \bar{Y}(k)(a+\frac{1}{\Delta t})^{-1} - \Delta Error(k) + N\bar{y}(k+1) + nf\bar{y}(k+1) > (\bar{Y}(k)-S(k))(a+\frac{1}{\Delta t})^{-1}.$$

Simplifing the inequality, we get

$$S(k) > \left[N\tilde{y}(k+1) + nf\tilde{y}(k+1) - \Delta Error(k)\right](a + \frac{1}{\Delta t})\text{, and}$$

$$-S(k) < \left[N\tilde{y}(k+1) + nf\tilde{y}(k+1) - \Delta Error(k)\right](a + \frac{1}{\Delta t})\,. \tag{43}$$

Since $S(k) > 0$, $-S(k) < \left[\sup_{\forall k>T_f}\{\|N\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|nf\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|\Delta Error(k)\|\}\right](a + \frac{1}{\Delta t})$ is always true. By assumptions 1, 2, and 3, the following inequalities will hold for the worst condition,

$$S(k) > \left[\sup_{\forall k>T_f}\{\|N\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|nf\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|\Delta Error(k)\|\}\right](a + \frac{1}{\Delta t})\,. \tag{44}$$

Apparently, $\left[\sup_{\forall k>T_f}\{\|N\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|nf\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|\Delta Error(k)\|\}\right](a + \frac{1}{\Delta t}) = \inf_{\forall k>T_f}\{S(k)\}$, which is the greatest lower bound of $S(k)$ and $-\left[\sup_{\forall k>T_f}\{\|N\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|nf\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|\Delta Error(k)\|\}\right](a + \frac{1}{\Delta t}) = \sup_{\forall k>T_f}\{-S(k)\}$, which is the least upper bound of $-S(k)$, and since $S(k) > S(k+1)$ and $-S(k) < -S(k+1)$, the following inequalities will always hold

$$\left[\sup_{\forall k>T_f}\{\|N\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|nf\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|\Delta Error(k)\|\}\right](a + \frac{1}{\Delta t}) \geq S(k+1)\text{, and}$$

$$-\left[\sup_{\forall k>T_f}\{\|N\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|nf\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|\Delta Error(k)\|\}\right](a + \frac{1}{\Delta t}) \leq -S(k+1)\,, \tag{45}$$

for both situations, $S(k+1) > 0$ and $S(k+1) < 0$, which implies

$$\Sigma \leq S(k+1) \leq \Xi\,, \tag{46}$$

For $S(k) < 0$:
Plugging Equations (42) and (10) into the inequality (3.9), we have

$$(\bar{Y}(k) - S(k))(a + \frac{1}{\Delta t})^{-1} > \bar{Y}(k)(a + \frac{1}{\Delta t})^{-1} - \Delta Error(k) + N\tilde{y}(k+1) + nf\tilde{y}(k+1) > (\bar{Y}(k) + S(k))(a + \frac{1}{\Delta t})^{-1}\,.$$

Simplifing the inequality, we get

$$-S(k) > \left[N\tilde{y}(k+1) + nf\tilde{y}(k+1) - \Delta Error(k)\right](a + \frac{1}{\Delta t})\text{, and}$$

$$S(k) < \left[N\tilde{y}(k+1) + nf\tilde{y}(k+1) - \Delta Error(k)\right](a + \frac{1}{\Delta t})\,. \tag{47}$$

Since $S(k) < 0$, $S(k) < \left[\sup_{\forall k>T_f}\{\|N\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|nf\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|\Delta Error(k)\|\}\right](a + \frac{1}{\Delta t})$ is always true. By assumptions 1, 2, and 3, the following inequalities will hold for the worst condition,

$$-S(k) > \left[\sup_{\forall k>T_f}\{\|N\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|nf\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|\Delta Error(k)\|\}\right](a+\frac{1}{\Delta t}). \qquad (48)$$

Apparently, $\left[\sup_{\forall k>T_f}\{\|N\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|nf\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|\Delta Error(k)\|\}\right](a+\frac{1}{\Delta t}) = \inf_{\forall k>T_f}\{-S(k)\}$, which is the greatest lower bound of $-S(k)$ and $-\left[\sup_{\forall k>T_f}\{\|N\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|nf\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|\Delta Error(k)\|\}\right](a+\frac{1}{\Delta t}) = \sup_{\forall k>T_f}\{S(k)\}$, which is the least upper bound of $S(k)$, and since $-S(k) > S(k+1)$ and $S(k) < -S(k+1)$, the following inequalities will always hold

$$\left[\sup_{\forall k>T_f}\{\|N\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|nf\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|\Delta Error(k)\|\}\right](a+\frac{1}{\Delta t}) \geq S(k+1), \text{ and}$$

$$-\left[\sup_{\forall k>T_f}\{\|N\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|nf\tilde{y}(k+1)\|\} + \sup_{\forall k>T_f}\{\|\Delta Error(k)\|\}\right](a+\frac{1}{\Delta t}) \leq -S(k+1), \qquad (49)$$

for both situations, $S(k+1) > 0$ and $S(k+1) < 0$, which implies

$$\Sigma \leq S(k+1) \leq \Xi, \qquad (50)$$

We get exactly the same result as Equation (46). Thus, the S function is bounded by the value defined by the least upper bounds of the remaining uncertainty of the nominal system, the residue of the on-line estimator, and the error by the optimization algorithm.

Q.E.D.

APPENDIX G:

Multiple Model Approach by Orthonormal Bases For Controller Design

by

Gary G. Yen and Seok-Beom Lee

Submitted to
International Journal of Control

MULTIPLE MODEL APPROACH BY ORTHONORMAL BASES FOR CONTROLLER DESIGN[*]

Gary G. Yen *Soek-Beom Lee*

Oklahoma State University
School of Electrical and Computer Engineering
Intelligent Systems and Control Laboratory
Stillwater, OK 74078

Abstract

Recently, model-free or data-driven control has been gaining great interest in overcoming the limitations of conventional model based control methodologies. However, the existing data-driven control is far from practical because of its slow convergence, severe computational burden, and lack of analysis and synthesis tools. This paper was motivated in response to these deficiencies of data-driven control methodologies. A multiple modeling method for subsequent robust controller design is proposed. This method is developed to overcome the difficulties of utilizing the existing multiple model approaches for control purpose by adopting a different locality concept based on dominant poles. The dominant poles are represented by Laguerre basis functions. Two examples are included to demonstrate the feasibility and characteristics of the proposed identification approach.

[*] This work was supported in part by the U.S. Air Force Office of Scientific Research under Grant F49620-98-1-0049 and National Science Foundation, Measurement and Control Engineering Center.

1. Introduction

When we are given a complex problem to solve, one of our instinctive approaches is to divide the problem into smaller and more manageable ones. In this spirit, multiple model control [Murray-Smith and Johansen, 1997] is an intuitive approach to solve complex control problems because it is originated from the same philosophy, the so called "divide and conquer strategy." From the control point of view, modeling is a very important procedure that has to be performed before designing controllers. In spite of the importance of modeling, there is no universal modeling method, therefore, it is not surprising that control engineers have to spend a good deal of time just to obtain a proper model for controller design. Another limitation of model based control is that model based control can only handle the environments considered beforehand. There are many cases in which a model cannot incorporate all possible scenarios such as unanticipated failure modes and structural autonomy. Adaptive control may be the solution for this case; however, most model based adaptive controllers can only handle a certain degree of uncertainties [Åström and Wittenmark, 1995].

To relieve the difficulties in modeling, linear system identification methods are available, however, linear models can represent a system only in small operating regimes. Neural networks have been suggested as a universal framework for nonlinear system identification, but the structures of neural networks are different from the ones that have been studied for nonlinear controller design. Also, training a neural network for an entire system under various operating conditions is not trivial, if not at all impossible. Bearing in mind these problems, multiple model control has been suggested as an alternative to handle a complex nonlinear system by decomposing it into more tractable subsystems and then integrating them together.

Even though the multiple model control mentioned above is concerned with the case where a global analytical model is not available, multiple model control can be also beneficial when a global nonlinear model is available. The reason is that nonlinear control design is still an evolving subject and the structures fit for nonlinear controller design are very restricted [Haber and Unbehauen, 1990]. In many cases, complex nonlinear systems can reduce to more tractable systems in local regions; hence local controller design for the local environment is relatively simpler than a global nonlinear controller design. Another motivation for multiple model control is that it can handle rapidly changing system dynamics or operating environments. Adaptive and robust control have also been suggested for these purposes and are under extensive investigation. However, switching, tuning, or interpolating action of multiple model control is expected to provide faster and more reliable control than other control methods [Narendra *et al.*, 1995].

Because the idea of multiple model control comes naturally, it is not surprising that multiple model control shares a common ground with gain scheduling [Apkarian and Adams, 1998], and hybrid systems [Antsaklis, 1998] to some extent. Also, similar approaches can be found in statistics and intelligent systems under different names such as local nonparametric method and lazy learning [Atkeson *et al.*, 1997a-b]. Compared to other control strategies, we use multiple model control as a general term to embrace all the similar methods.

As mentioned previously, multiple model control finds its benefits whether a theoretical model is available or not. However, the multiple model control approach with available models is usually problem dependent and will be the subject of nonlinear control theory [Kaplan and

Glass, 1995]. Therefore, this study is concerned with the cases *without a proper theoretical model* and is limited to *empirical modeling process and control.* Even though the ideas and expected benefits of multiple model control are natural and attractive, there are many technical details that have to be resolved to make multiple model control practical. The following is a brief list of the issues of empirical multiple model control that have to be resolved:

1. defining locality (how to distribute or partition a system);
2. identification of subsystems;
3. switching, interpolation, and extrapolation strategy;
4. local controller design and validation;
5. realizing global stability and performance;
6. on-line adaptation of models and controllers; and
7. implementation issues.

The first three items are regarding modeling. Validation of local models is necessary to verify the local models and estimate local uncertainties for robustness of the subsystems. The second three items are regarding analysis and synthesis of multiple model based control. The implementation issues such as computational complexity and the effect of quantization error must be addressed to realize a practical multiple model control.

As a first stage in achieving practical multiple model control, this paper emphasizes the modeling part. The main concerns are the empirical modeling from observed data by forming subsystems and coordinating them to achieve a good global model with estimated quality description. Most multiple model techniques lack rigorous analysis of model quality. For robust controller design, which is inevitable for data-driven control because of model error caused by noisy and biased data, model quality description with limited complexity of nominal model is essential. In Section 2, a multiple modeling algorithm based on orthonormal bases and uncertainty estimation is developed to overcome the problems of the existing multiple model approaches to controller design. To demonstrate the use and characteristics of existing methods and the new development, simulation study is included in Section 3. The simulation study shows the promising results of multiple modeling approaches. In Section 4, conclusions are drawn in providing relevant observations and future research directions.

2. Multiple Model Approach by Orthonormal Bases

Multiple modeling may be seen as a branch of nonlinear system identification. Comparable to global methods such as neural networks, multiple modeling divides a system into subsystems. One driving force of decomposition is so called temporal crosstalk [Jordan and Jacobs, 1994], which makes once trained global models forget what was learned previously. Also, the identification of subsystems is more efficient and transparent than with global methods. The heart of multiple modeling is to divide the system, identify local models and then combine them. Approaches to multiple modeling can be roughly divided into two vategories: probabilistic and non-probabilistic. Non-probabilistic approaches [Narendra, 1996], [Narendra and Mukhopadhyay, 1997], [Principe *et al*, 1998] are based on the prediction error or the geometric information of data while probabilistic approaches [Anderson, 1985], [Kadirkamanathan and Fabri, 1998] are based on the assumed probability of the system.

As seen from extensive literature reviews, several multiple modeling methods have been proposed as alternatives to model complex nonlinear systems. However, a short glimpse of the proposed methods soon reveals that they lack a theme, that is, the purpose of the models. To remind us that a model is merely a suitable description of representing physical phenomena, models without designated purpose can hardly be useful in practice. In this study, the focus is on developing models for designing robust controllers. By putting this purpose in a modeling process, our model has to have certain constraints: limited complexity and explicit representation of uncertainty of the model. Because we are interested in data driven modeling or system identification, uncertainty representation should also depend on estimation instead of derivation from physical interpretations. Because of several benefits of orthonormal basis, orthonormal basis functions are usually used for system identification for robust control [Bodin *et al.*, 1997]. A review of representative orthonormal basis function in literature is given first. Motivated from the reviewed uncertainty estimation techniques and orthonormal basis functions, a new multiple modeling method is then proposed.

2.1 Orthonormal bases

As seen from most nonlinear system identification framework, a regression vector usually consists of a series of past inputs and outputs. However, this may cause propagation of estimation error and sometimes causes instability of models [Nelles, 1998]. For this reason, only input dependent regressors are attractive. The problem with only input dependent regressors such as the FIR model is that they require very high order models to achieve a satisfactory result. For this reason, filtered inputs instead of pure inputs can be used to reduce the complexity of the models. Recently, orthonormal basis functions have attracted a great interest. By using orthonormal basis functions, parameter estimation is treated as a well-known linear least square approximation. So, it accelerates the estimation of parameters, avoids the convergence to local minimum, and facilitates the analysis of the model properties. In [Wahlberg, 1994], an orthonormal basis function is derived by an optimization solution in the sense of *n-width* measure. The *n-width* measures the smallest approximation error for the worst possible system using the best possible n-dimensional linear-in-the-parameters models set. The n-width measure is formally defined as [Tøffner-Clausen, 1996]:

Definition 1 (*n-width* measures)
Assume that we know that the system G belongs to a given bounded set S, $G \in S$, then *n-width measure* is defined as:

$$d_n(S;B) = \inf_{\Phi_n \in M_n(B)} \sup_{G \in S} \inf_{G_n \in \Phi_n} \|G - G_n\|_B \tag{1}$$

where B denotes some Banach space with norm $\|\cdot\|_B$, e.g., H_2 or H_∞, $M_n(B)$ denotes the collection of all n-dimensional linear subspaces of B, Ψ_j spans Φ_n and $G_n = \sum_{j=1}^{n} g_j \Psi_j$. S is a priori given bounded set, which is $G \in S$. The innermost term $e_n^\Phi(G) = \inf_{G_n \in \Phi_n} \|G - G_n\|_B$ denotes the best achievable model errors for G by elements in Φ_n. Let $\Phi = \{\Phi_n\}_{n \geq 1}$ denotes the corresponding sequence of subspaces. Φ is called complete if for any $G \in B$, $e_n^\Phi(G) \to 0$ as

$n \to \infty$. If $d_n(S;B) = \sup_{G\in S} \inf_{G_n \in \Phi_n^*} \|G - G_n\|_B$, then Φ_n^* is called an optimal subspace for $d_n(S;B)$.

For exponentially stable systems (analytic in $|z| > R$), *FIR model* are proven to be optimal in the *n-width* sense for H_2 and H_∞ [Pinkus, 1985]. However, if the information about the dominating poles is available, then the *Laguerre basis function* is proven to be optimal for systems $G(s) \in H_2(C,C)$, which are analytic outside the domain $|s+\alpha| \le r, \alpha > r$ [Wahlberg, 1994]. Here in $H_2(\cdot,\cdot)$, the first argument is the domain of $G(\cdot)$ and the second is the range. This corresponds to a situation where we know the dominating poles of the system belong to the set $|s+\alpha| \le r$. Also, it is proven that the space spanned by the Kautz functions, $k = 1,\cdots,n$ with $b = \sqrt{\alpha^2 - r^2}$, is an optimal $2n$-dimensional subspace in the *n-width* sense for function $G(s) \in H_2(C,C)$, which are analytic outside the domain $|(s^2 + \alpha s + c)/2| \le r, r < \alpha$ [Wahlberg, 1994]. Hence, Laguerre basis function is often used to approximate well-damped systems while Kautz basis function is used for approximating lightly damped systems. Therefore, the combination of Laguerre and Kautz basis functions can approximate any linear systems very well. The following theorem for continuous-time Laguerre functions is cited from [Wahlberg, 1994].

Theorem 1 (*n-width* of continuous-time Laguerre models)
The space Φ_n^* spanned by the *Laguerre functions*

$$L_j(s,a) = \frac{\sqrt{2a}}{s+a}\left[\frac{s-a}{s+a}\right]^{j-1}, \quad j = 1,\cdots,n \tag{2}$$

with *Laguerre pole*, $a = \sqrt{\alpha^2 - r^2}$ is an optimal n-dimensional subspace in the *n-width* sense for functions $G(s) \in H_2(C,C)$, which are analytic outside the domain $|s+\alpha| \le r, \alpha > r$.

In addition, there is a corresponding theorem for discrete-time Laguerre functions [Tøffner-Clausen, 1996].

Theorem 2 (*n-width* of discrete-time Laguerre models)
The space Φ_n^* spanned by the *discrete-time Laguerre basis functions*

$$L_j(z,a) = \frac{\sqrt{1-a^2}}{z-a}\left[\frac{1-az}{z-a}\right]^{j-1}, \quad j = 1,\cdots,n, |a| < 1 \tag{3}$$

with Laguerre pole

$$a = \frac{1}{2\alpha}(1 + \alpha^2 - r^2 + \sqrt{(1+\alpha^2 - r^2)^2 - 4\alpha^2}) \tag{4}$$

is an optimal n-dimensional subspace in the *n-width* sense for functions $G(z) \in H_2(C,C)$ which are analytic outside of the domain $|z-\alpha| \le r$ with $r > |\alpha| + 1$.

The realization of Laguerre basis functions is represented as an expansion as $y(t) = \sum_{i=1}^{m} \theta_i L_i(z,a)u(t)$, which can be considered as filtered inputs through a low pass filter and a sequence of all pass filters (as shown in Figure 1). The problem of using these orthonormal basis functions is that it requires a prior knowledge about the system poles. A crude estimation can be obtained from step or impulse response of the system or from a first order ARX model. More optimal poles can be obtained from numerical search [Malti *et al.*, 1998]. In [Oliveira e Silva, 1995], Taylor series are used to find the optimal poles by utilizing a certain property of Laguerre filters in optimal poles. Study regarding more general orthonormal filters can be found in [Bodin *et al.*, 1997].

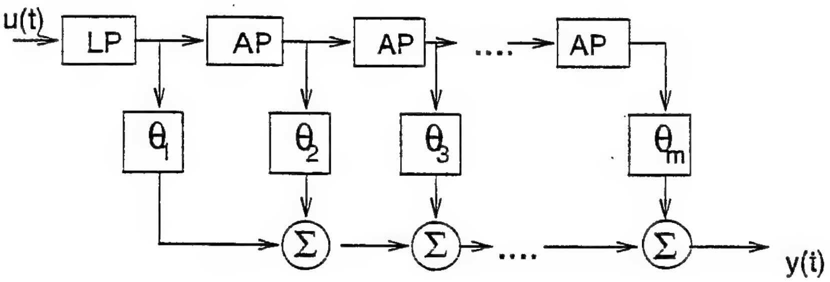


Figure 1: Laguerre network.
(LP denotes a first order low-pass while AP denotes a first order all-pass filter.)

2.2 Nonlinear system identification with Laguerre bases

As pointed out earlier, the existing multiple modeling methods lack quality descriptions. As a matter of fact, deciding the interpolating or switching variables in the multiple models is a relatively straightforward task once local models are identified. For example, for a single-input-single-output system, the multiple model may be written as the following:

$$y = \sum_{i=1}^{m} w_i \hat{y}_i \tag{5}$$

where $\hat{y}_i$ and w_i are a local model output and a weight for the *i*th model, respectively. As can be easily seen, the weights are linear in parameter and can be easily updated by recursive least square or Kalman filter. Therefore, the more demanding task in multiple modeling is identification of local models.

Intuitively, it is beneficial to have a constructive method to define locality of the models. The literature review of orthonormal functions such as Laguerre basis functions reveals that they are optimal in approximating a linear system with certain knowledge of pole locations. Also, nonlinear systems can be approximated by linear systems in a local sense. Hence, it is natural to explore orthonormal basis functions for nonlinear system identification with time varying or system dependent parameters. Another motivation of using orthonormal basis functions is that the model maintains the linear-in-parameter property. This makes the analysis and synthesis of the model tractable such that the uncertainty bounds for robust control can be estimated from the bias and variance information of the estimated parameters.

In this section, an innovative algorithm is proposed which utilizes multiple Laguerre bases for nonlinear system identification. The issues resolved are the selection of pole locations for each Laguerre basis and parameter estimation. For simplicity, the algorithm is limited to single-input single-output systems. The initial thought was to select the Laguerre poles to be orthonormal to each other such that each local model spanned by each Laguerre basis become orthonormal. By this, the locality of local models does not depend on distance metric in regression space or on prediction errors. For example, consider the following system:

$$y(t) = f(\phi(t)) + v(t)$$

where $y(t)$, $v(t)$ and $\phi(t)$ are output, measurement noise and regression vector, respectively. The model by multiple Laguerre bases can be written as the following:

$$\begin{aligned}\hat{y}(t) &= (\Lambda^1(q)\theta^1 + \Lambda^2(q)\theta^2 + \cdots + \Lambda^n(q)\theta^n)u(t) \\ &= \phi^1(t)\theta^1 + \phi^2(t)\theta^2 + \cdots + \phi^n(t)\theta^n \\ &= \hat{y}^1(t) + \hat{y}^2(t) + \cdots + \hat{y}^n(t)\end{aligned}$$

where Λ^j is a local model spanned by *j*th Laguerre basis, ϕ^j is a local regression vector. Each local model is expanded as $\hat{y}^j(t) = \sum_{i=1}^{m} \Lambda_i^j(q)\theta_i^j u(t)$.

Then each local model can be identified simply by refitting residuals to least square estimation until the desired accuracy is achieved. This process holds many advantages over the existing multiple model approaches: (1) a constructive modeling process to decide the number of local models and local model structures, (2) natural estimation of global uncertainty bounds by $\hat{y}(t) = \{\Lambda^1(q)(\theta^1 + \Delta^1) + \Lambda^2(q)(\theta^2 + \Delta^2) + \cdots + \Lambda^m(q)(\theta^m + \Delta^m) + \Delta(q)\}u(t)$ where Δ^i is a real vector from covariance of parameter estimation and $\Delta(q)$ is a complex function from undermodeling. However, it soon turned out that selecting Laguerre poles to be orthonormal to each other is not a trivial process without knowledge of system poles. Therefore, a different, somewhat heuristic, approach is used in this study.

Since we do not use any assumptions about the system, the poles must be estimated from the data. For the optimal pole locations, a numerical search based on the Newton-Raphson method is adopted. The derivatives can be computed in closed forms because of the nice property of Laguerre function. The derivation is similar to the continuous time case in [Malti *et al.*, 1998], but is extended into the discrete-time Laguerre basis here. Consider a local Laguerre basis function expansion:

$$\hat{y}^j(t) = G^j(q)u(t) = \sum_{i=1}^{m} \Lambda_i^j(q)\theta_i^j u(t) \qquad (6)$$

where $\Lambda_i^j(q) = \dfrac{\sqrt{1-(a^j)^2}}{q-a^j}\left[\dfrac{1-a^j q}{q-a^j}\right]^{i-1}$, $i = 1,\cdots,m$ and $\left|a^j\right| < 1$. a^j is the pole of the *j*th local Laguerre basis set. Define a cost function to be the following:

$$J = \frac{1}{2}e'e \tag{7}$$

where $e = Y - X\theta^j$, $\theta^j = \left[\theta_1^j \quad \theta_2^j \quad \cdots \quad \theta_m^j\right]'$, $Y = \left[y(1) \quad y(2) \quad \cdots \quad y(N)\right]'$, and $X = \left[u(1) \quad u(2) \quad \cdots \quad u(N)\right]'\left[\Lambda_1^j(q) \quad \Lambda_2^j(q) \quad \cdots \quad \Lambda_m^j(q)\right] = U\left[\Lambda_1^j(q) \quad \Lambda_2^j(q) \quad \cdots \quad \Lambda_m^j(q)\right]$.

The Newton-Raphson method to estimate local Laguerre pole is given as:

$$a_{n+1}^j = a_n^j - \mu\left(\frac{\partial^2 J(a_n^j)}{\partial (a^j)^2}\right)^{-1}\frac{\partial J(a_n^j)}{\partial a^j} \tag{8}$$

where $\partial J/\partial a^j$ and $\partial^2 J/\partial (a^j)^2$ are evaluated at a_n^j, μ is the step size. The derivative of *J* with respect to a^j is the following:

$$\frac{\partial J}{\partial a^j} = e'\frac{\partial e}{\partial a^j} = e'\left(-\frac{\partial X}{\partial a^j}\theta^j - X\frac{\partial \theta^j}{\partial a^j}\right). \tag{9}$$

The parameter vector θ^j can be written in a closed form solution, the so called normal equation:

$$\theta^j = (X'X)^{-1}X'Y. \tag{10}$$

The partial derivative of the normal equation can be written as:

$$\frac{\partial \theta^j}{\partial a^j} = \left(X'X\right)^{-1}\left[\frac{\partial X'}{\partial a^j}Y - \left(\frac{\partial X'}{\partial a^j}X + X'\frac{\partial X}{\partial a^j}\right)\theta^j\right]. \tag{11}$$

The second term of (29), $e'X\dfrac{\partial \theta^j}{\partial a^j}$, becomes zero since

$$\theta^j = (X'X)^{-1}X'Y = (X'X)^{-1}X'(X\theta^j + e) = \theta^j + (X'X)^{-1}X'e.$$

Hence, $(X'X)^{-1}X'e = 0$ and from (11), (9) becomes the following:

$$\frac{\partial J}{\partial a^j} = -e'\frac{\partial X}{\partial a^j}\theta^j. \tag{12}$$

The second derivative of *J* with respect to a^j is:

$$\frac{\partial^2 J}{\partial (a^j)^2} = \left((\theta^j)^t \frac{\partial X^t}{\partial a^j} + \frac{\partial (\theta^j)^t}{\partial a^j} X^t \right) \frac{\partial X}{\partial a^j} \theta^j - e^t \left(\frac{\partial^2 X}{\partial (a^j)^2} \theta^j + \frac{\partial X}{\partial a^j} \frac{\partial \theta^j}{\partial a^j} \right). \tag{13}$$

So, the remaining equations to be derived are the first and second partial derivatives of *X*. The derivation is done by the following Laguerre filter property:

$$\frac{\partial \Lambda_i^j}{\partial a^j} = \frac{i\Lambda_{i+1}^j (a^j) - (i-1)\Lambda_{i-1}^j (a^j)}{1-(a^j)^2}, \quad i = 1, \cdots, m. \tag{14}$$

Therefore, the first partial derivative of *X* is:

$$\frac{\partial X}{\partial a^j}(:,i) = U \left(\frac{i\Lambda_{i+1}^j (a^j) - (i-1)\Lambda_{i-1}^j (a^j)}{1-(a^j)^2} \right), \quad i = 1, \cdots, m, \tag{15}$$

where $(:,i)$ means the *i*th column of a matrix. The second partial derivative of *X* is the following:

$$\frac{\partial^2 X}{\partial (a^j)^2}(:,i) = U \left(\begin{array}{l} \dfrac{i(i+1)\Lambda_{i+2}^j (a^j) + 2a^j i\Lambda_{i+1}^j (a^j) - (2i^2 - 2i + 1)\Lambda_i^j (a^j)}{(1-(a^j)^2)^2} \\ + \dfrac{-2a^j (i-1)\Lambda_{i-1}^j (a^j) + (i-1)(i-2)\Lambda_{i-2}^j (a^j)}{(1-(a^j)^2)^2} \end{array} \right), \quad i = 1, \cdots, m. \tag{16}$$

The algorithm of estimating the optimal pole involves estimating parameters by (10) and then to advancing pole estimation using (8), (12), (13), (11), (15), and (16). This process can be iterated until an optimal solution is obtained. As well known, optimization problems can get stuck in local minimum, therefore, the optimization must be repeated with different initial conditions. This algorithm is implemented in a Matlab function in this study. Using the optimization algorithm derived above, Laguerre poles can be located. As mentioned before, selection of Laguerre poles orthonormal to each other is not a trivial problem. So, a set of training data is divided into multiple sets of local data sets, then the derived optimization algorithm in the above is applied. If the estimated poles at different data sets are different enough, these poles are considered to be the poles for local Laguerre bases. This procedure is somewhat heuristic, however, optimization becomes very efficient since optimization is performed only in a single dimension with a small data set. A moving window along time horizon can also be used instead of segmenting data set.

2.3 Parameter estimation by recursive least square with forgetting

Since the multiple models are linearly parameterized, the estimation of parameters is very efficient. In this study, recursive least square with a forgetting factor is adopted [Anderson, 1985]. This is based on the assumption that a nonlinear system can be approximated by a linear system with time varying parameters. Therefore, the parameters must be adapted with time or states. The well-known recursive least square with forgetting factor is given as:

$$\theta_{k+1} = \theta_k + P_k X(k+1)\alpha_k \left(y(k+1) - X(k+1)' \theta_k \right) \tag{17}$$

where $\alpha_k = \frac{1}{\lambda + X(k+1)' P_k X(k+1)}$. λ is a forgetting factor, $X(k+1)$ is a regression vector at $k+1$ sequence, $y(k+1)$ is a output at $k+1$ sequence and α_k is the estimated parameter at k sequence. P_k is a initially large number and is recursive updated by $P_{k+1} = \frac{P_k - P_k X(k+1)\alpha_k^{-1} X(k+1)' P_k}{\lambda}$.

2.4 The proposed identification algorithm

The proposed algorithm consists of two stages: (1) off-line local Laguerre pole estimation by the algorithm derived in Subsection 2.2, and (2) on-line recursive parameter estimation shown in Subsection 2.3. The following procedure shows the identification processes involved:

1. segment the data set
2. find a Laguerre pole by using the optimization algorithm using (10), (8), (12), (13), (11), (15), and (16).
3. if the identified pole is different enough, accept it as an additional system pole
4. repeat the above process with the next segmented data set
5. after identifying the poles, apply the parameter estimation algorithm on-line using (17).

3. Simulation Study

This section is intended to verify the characteristics of the proposed algorithm for system identification by simulation study. Two discrete time models are assumed to be the true systems that generated data: (i) a SISO linear system, (ii) a SISO nonlinear system. They are quoted from a published paper [Stenman, 1999]. For more realistic simulations, outputs are corrupted with Gaussian random noise. The selected systems are listed in the following.

(i) SISO linear system:

$$y(t) - 1.5y(t-1) + 0.7y(t-2) = u(t-1) + 0.5u(t-2) + \varepsilon(t); \tag{18}$$

(ii) SISO nonlinear system:

$$\begin{aligned}
x_1(t+1) &= \left(\frac{x_1(t)}{1 + x_1^2(t)} + 1 \right) \sin(x_2(t)) \\
x_2(t+1) &= x_2(t)\cos(x_2(t)) + x_1(t)\exp\left(-\frac{x_1^2(t) + x_2^2(t)}{8} \right) + \frac{u^3(t)}{1 + u^2(t) + 0.5\cos(x_1(t) + x_2(t))} \\
y(y) &= \frac{x_1(t)}{1 + 0.5\sin(x_2(t))} + \frac{x_2(t)}{1 + 0.5\sin(x_1(t))} + \varepsilon(t).
\end{aligned} \tag{19}$$

The models considered for system identification are the following four: (1) linear AutoRegression with eXternal inputs (ARX) model, (2) Feedforward neural networks (FFNN) based ARX model [Narendra *et al.*, 1995], (3) Self-Organizing Map (SOM) oriented multiple model [Principe *et al.*, 1998], and (4) the multiple model with multiple Laguerre bases proposed.

3.1 Identification of a linear discrete-time system

In order to generate data, the system (i) was simulated with $u(t)$ and $\varepsilon(t)$ selected as independent and Gaussian distributed sequences with zero means and unit variances. The preliminary regression order selection algorithm based on Lipschitz quotients is applied to the data [He and Asada, 1993]. The algorithm is written as a function in Matlab. The function returns the index vector to be used for order estimation in the range of specification. This function requires extensive computation. Hence, an interactive function is also written to compute Lipschitz quotients only at user specified orders.

The results can be seen in Figure 2. The numbers in the bracket means input delay, order of input regression vector, and order of the output regression vector. In other words, $\lfloor n_k \;\; n_u \;\; n_y \rfloor$ for a model $y(t) = f\big(y(t-1),\cdots,y(t-n_y),u(t-n_k),\cdots,u(t-n_u)\big)$. From the graph, we can see that the curve converges after $\lfloor 1\ 2\ 2 \rfloor$. Hence, it will be a reasonable estimate of the regression order.

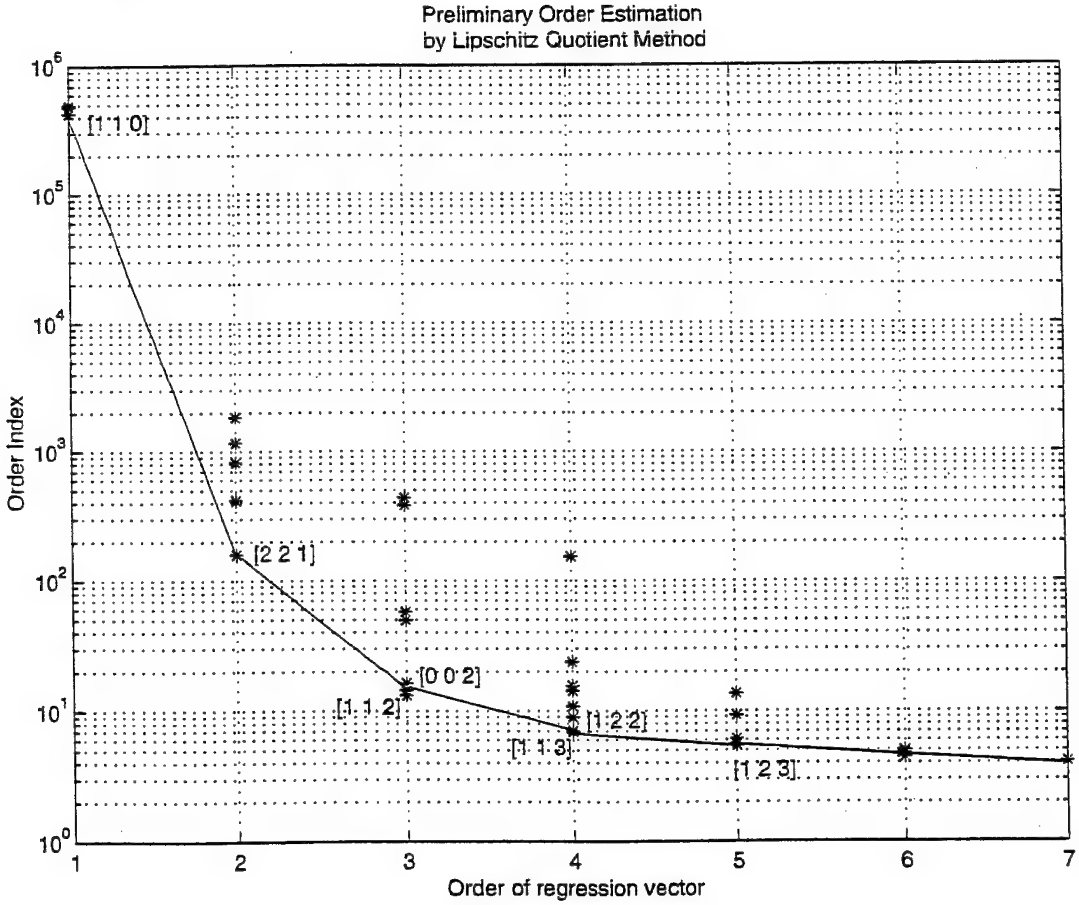
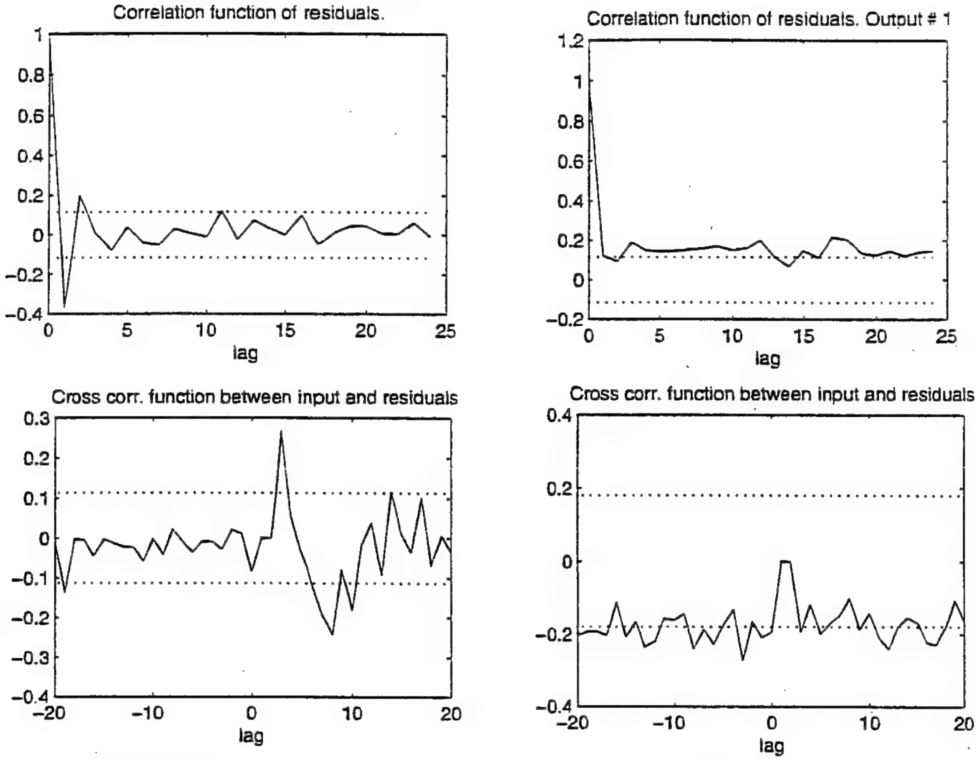


Figure 2: Order estimation by Lipschitz quotients for system (i)

It is well known that pre-processing training data can help the identification process. However, it is noticed that crude normalization can actually damage the quality of model. In Figure 3, the adverse effect of normalization of data is shown. Two data sets, one normalized by $\left(x-\frac{\min x+\max x}{2}\right)/\left(\frac{\max x-\min x}{2}\right)$ and the other not normalized, are used to identify the system with linear ARX models. The correlation of residual shows that estimation is biased because of the normalization. Also, the adverse effect of normalization seems significant for non-white noise cases. As a result, no normalization was applied for ARX models.



(a) output error system

(b) ARX system

Figure 3: The effect of normalization of data

For Self-Organizing Map (SOM) based multiple modeling, normalization is essential because SOM depends on the geometric property of data points. Without normalization, SOM does not cover the regression space well. The designs of all four models considered are realized off-line. The normalized data is used for the feedforward neural networks (FFNN) based ARX model and the SOM based multiple model while the raw data set is used for the ARX and multiple models by multiple Laguerre bases. Training of the ARX model was very efficient while the FFNN and SOM models required several trial and error to find reasonable models. It was surprising that the FFNN model was no better than other models in spite of the general belief that neural networks are good at approximating functions [Narendra and Parthasarathy, 1990]. The design parameters such as the number of hidden neurons in neural networks, the number of neurons in the SOM model and the number of Laguerre poles in the proposed method are all selected ad hoc since there are no existing design guidelines. The simulated response of each model with the training data set is shown in Figure 4 and with the validation data set in Figure 5. Average RMS values of simulated errors for 10 runs on each model are listed in Table 1. Notice that the proposed algorithm produced competitive performance compared to other three models.

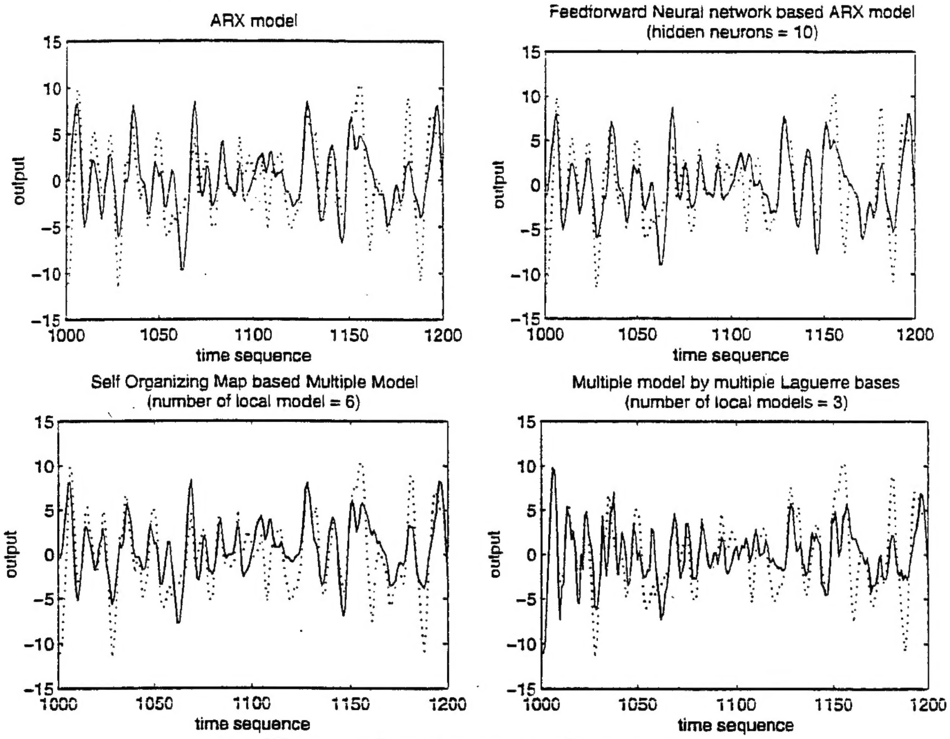


Figure 4: Training results of system (i)
(dotted line: output of training data, solid line: simulated output)

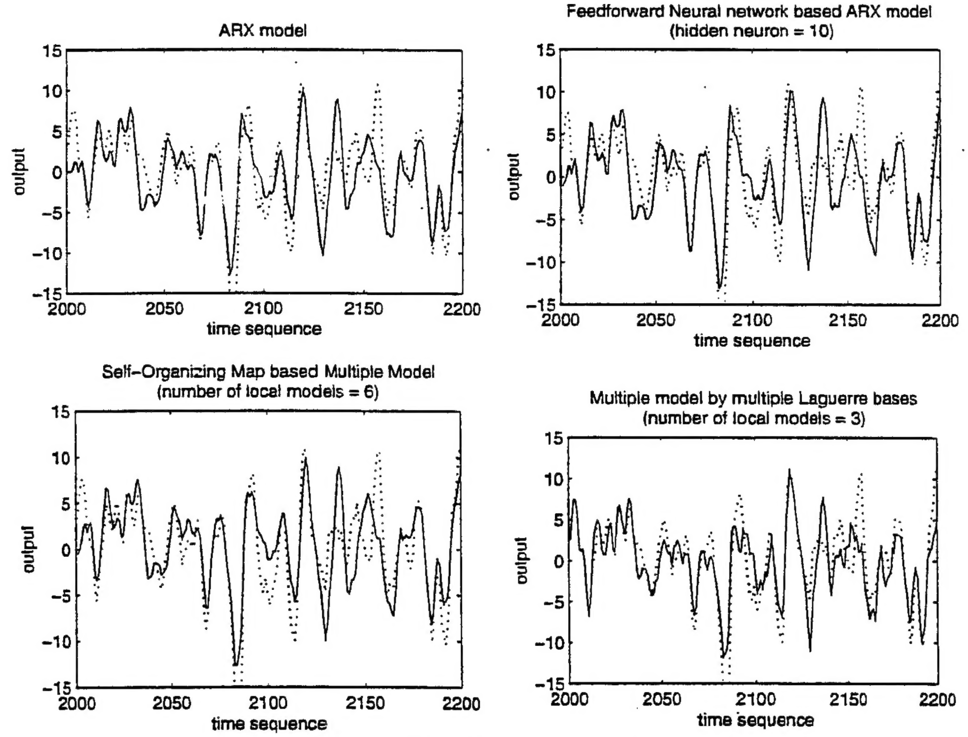


Figure 5: Generalization results of system (i)
(dotted line: output of training data, solid line: simulated output)

Table 1:RMS error of each model for system (i)

Model	training RMS error	generalization RMS error
ARX model	2.8030	3.2258
Neural network based ARX model	2.7365	3.3841
Self-Organizing map based multiple model	2.8726	3.5082
Multiple model by multiple Laguerre bases	2.7861	3.2256

3.2 Identification of a nonlinear discrete time system

After the assuring results of system (i) that these algorithms can identify the unknown system to a certain extent, a more complex and nonlinear system is tested. The data is collected with the input ranging from -2.5 to 2.5 and measurement noise of variance 0.1. The procedure of identification was similar to system (i). The preliminary order estimation method by Lipschitz quotients is applied and the result is shown in Figure 6. The graph shows that [1 3 2] may be a reasonable choice.

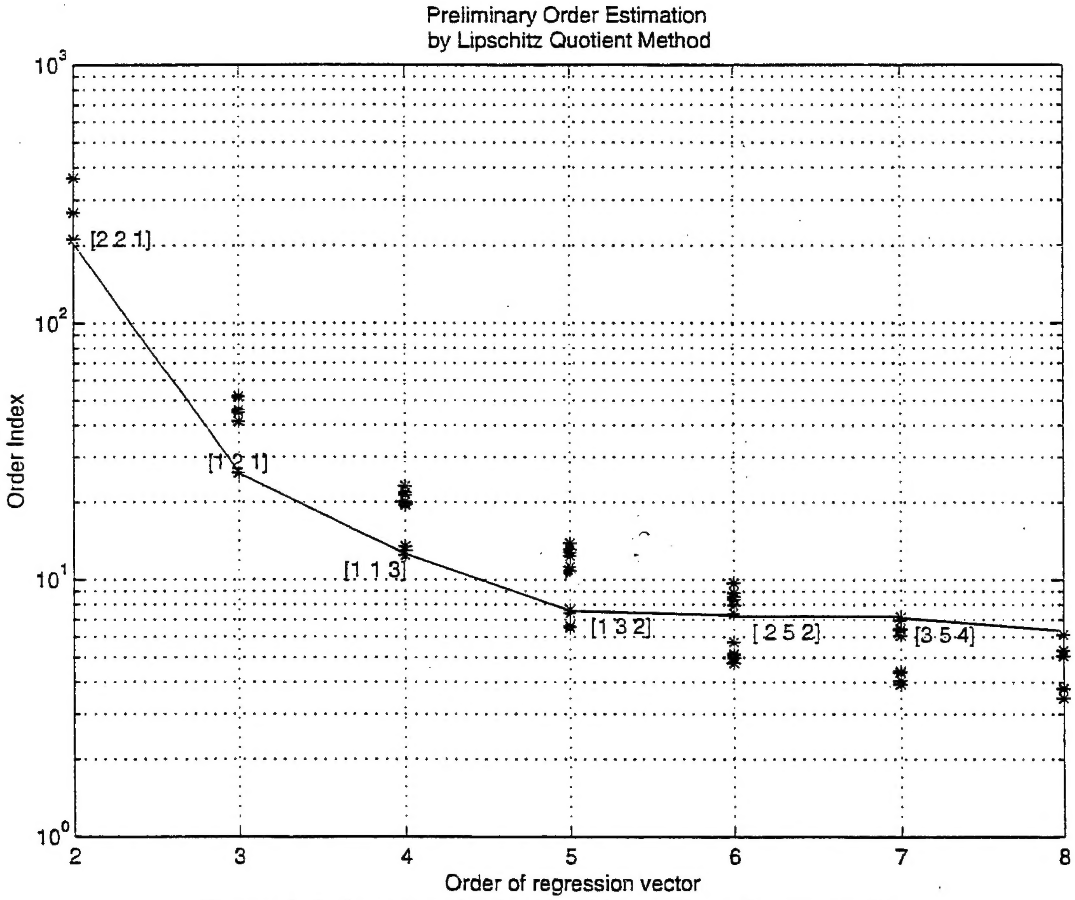


Figure 6: Order estimation by Lipschitz quotients for system (ii)

Similar to the system (i), training of the linear ARX model was very efficient. The training of the neural network model was not slow by using the efficient Levenberg-Marquardt algorithm, however, the response was no better than that of the linear ARX model and was worse in generalization. In contrast to the disappointing results of a global technique, multiple model approaches were efficient in training and the generalization results were excellent. Average RMS values of simulated errors for 10 runs on each model are listed in Table 2. As shown in Table 2, the proposed method produced the smallest error in the training and generalization sets. In Figures 7 and 8, the results from training set and generalization set are shown.

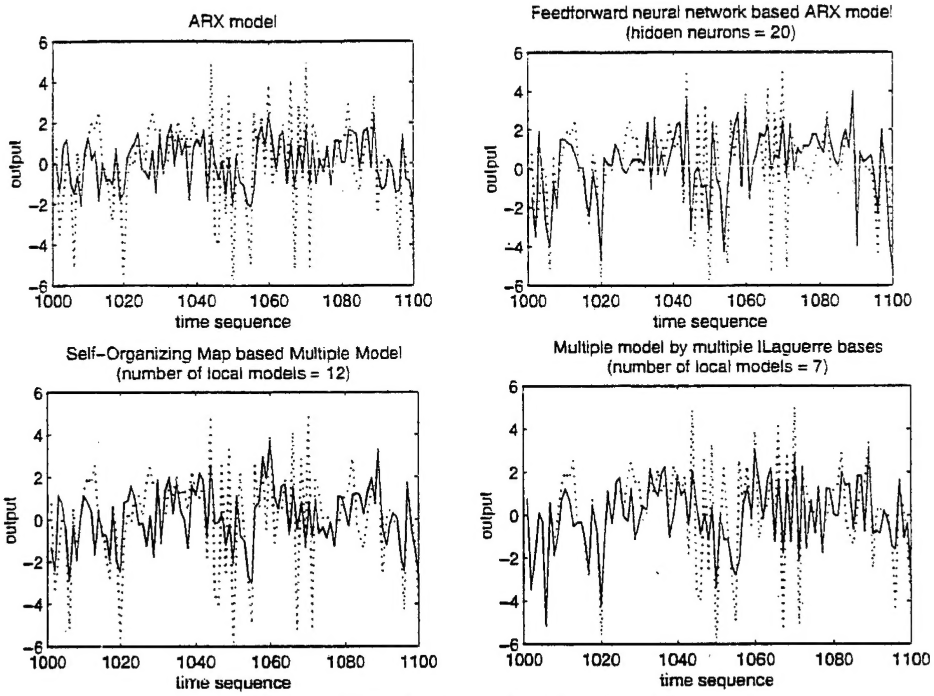


Figure 7: Training results of system (ii)
(dotted line: output of training data, solid line: simulated output)

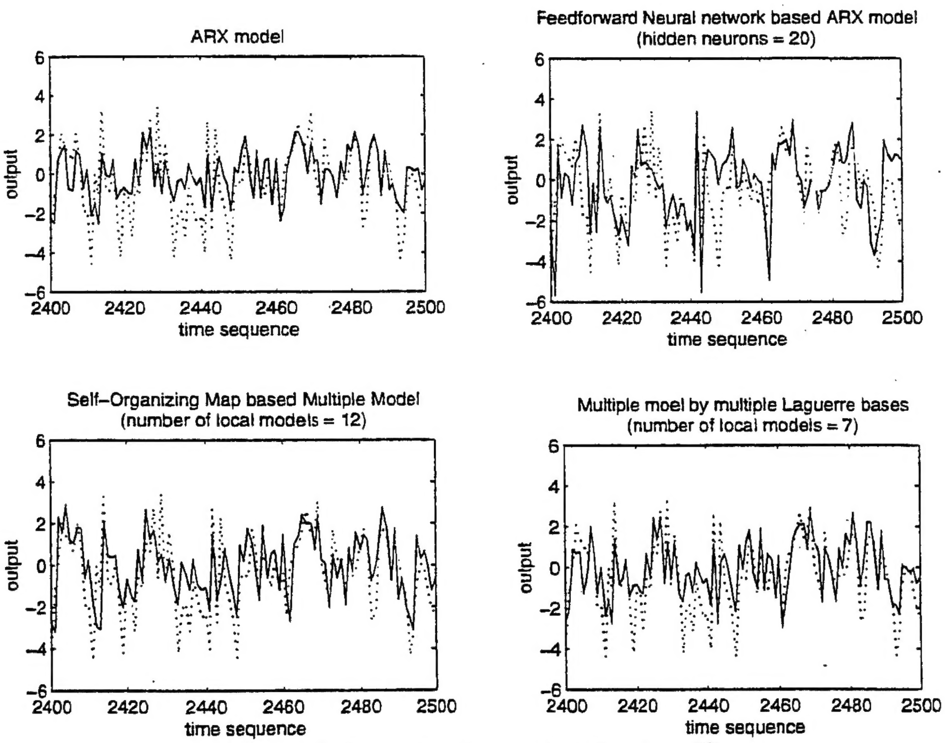


Figure 8: Generalization results of system (ii)
(dotted line: output of training data, solid line: simulated output)

Table 2:RMS error of each model for system (ii)

Model	training RMS error	generalization RMS error
ARX model	1.6322	1.6292
Neural network based ARX model	1.6956	1.9823
Self-Organizing map based multiple model	1.5675	1.7222
Multiple model by multiple Laguerre bases	1.5024	1.5074

4. Conclusion

This study was motivated by the vision to realize a practical data-driven control system comparable to conventional model based methods. The main theme was to adopt a multiple model approach instead of a global one. By this adoption, we can relieve the computational burden significantly as well as maintain the mathematical tractability. Also, this approach enables us to take advantage of the existing methods such as linear system identification and many of estimation techniques.

The proposed algorithm takes advantage of the properties of orthonormal basis functions which resulted in simple derivation of Laguerre pole estimation. By this, we can relieve the difficulty of estimating the order of regression vectors and also achieve efficient training with maintained mathematical tractability. The simulation results demonstrate the characteristics of the algorithm. Three other models were also considered: linear ARX model, feedforward neural networks based ARX model and SOM based multiple models. Even though the simulation results do not demonstrate significant improvement over other identification methods, the proposed identification algorithm will be truly beneficial when it is used in connection with robust controller design. The argument is well supported since there is a great interest in robust control community in developing robust control theory for linear time variant systems. The proposed identification model will be ideal for robust controller design since it will preserve proper characteristics for robust control such as reasonable model complexity and facilitation of estimating model error bounds.

References

Anderson, P., "Adaptive forgetting in recursive identification through multiple models," *International Journal of Control*, **42**, pp. 1175-1193, 1985.

Antsaklis, P. J., "Guest Editorial Hybrid Control Systems: an introductory discussion to the special issue," *IEEE Transactions on Automatic Control*, **43**, pp. 457-459, 1998.

Apkarian, P. and Adams, R. J., "Advanced gain-scheduling techniques for uncertain systems," *IEEE Transactions on Control Systems Technology*, **6**, pp. 21-32, 1998.

Åström, K. J. and Wittenmark, B., *Adaptive Control*, Addison-Wesley: Menlo Park, CA, 1995.

Atkeson, C. G., Moore, A. W. and Schaal, S., "Locally weighted learning," *Artificial Intelligence Review*, **11**, pp. 11-73,1997.

Atkeson, C. G., Moore, A. W. and Schaal, S., "Locally weighted learning for control," *Artificial Intelligence Review*, **11**, pp. 75-113, 1997.

Bodin, P., Villemoes, L. F. and Wahlberg, B., "An algorithm for selection of best orthonormal rational basis," *Proceedings of IEEE Conference on Decision and Control*, San Diego, California, pp. 1277-1282, 1997.